Defense Information Infrastructure (DII)
Common Operating Environment (COE)

How To Segment Guide

Version 4.0

30 December 1996

Prepared by:

Joint Interoperability and Engineering Organization
Defense Information Systems Agency

Table Of Contents

Paragraph **Page**

FORWARD

This document will be reviewed and updated by the Defense Information Systems Agency (DISA) as required to remain current with technology and program requirements. This document supersedes all previous Defense Information Infrastructure (DII) Common Operating Environment (COE) How To Segment Guide documents.

Changes to this document must be approved by DISA, but comments and recommendations for change may be forwarded for review and incorporation to:

DISA/JIEO
Center for Computer Systems Engineering
DII Computer Engineering Department
Attention: Mrs. Jeanne L. Muenzen
5600 Columbia Pike
Falls Church, Virginia 22041
email: muenzenj@ncr.disa.mil

Table Of Contents

Paragraph **Page**

Table Of Contents

Paragraph **Page**

Table Of Contents

Paragraph **Page**

Table Of Figures

Figure **Page**

Table Of Tables

Table **Page**

1. Introduction

The Defense Information Infrastructure (DII) Common Operating Environment (COE) serves as a foundation for building multiple systems. The DII COE should be considered as a "plug and play" open architecture designed around a client/server model. Functionality is easily added to or removed from the target system in small manageable units, called segments. Segments are defined in terms of functions that are meaningful to users, not in terms of internal software structure. Structuring the software into segments in this manner is a powerful concept that allows considerable flexibility in configuring the system to meet specific mission needs or to minimize hardware requirements for an operational site. Site personnel perform field updates by replacing affected segments through the use of a simple, consistent, graphically oriented user interface.

1.1 Purpose

The purpose of this document is to provide an application software developer the technical steps necessary to segment a software application for the DII COE. The segmentation process can be used for candidate DII COE components as well as mission applications which execute on top of the DII COE.

1.2 Scope

The scope of this document covers the developer's process from ordering and installing the DII COE and Developer's Toolkit to submitting the segment to DISA for integration and testing. The focus of the segmentation process described here is on generating the segmentation descriptor files and organizing development files into a structure that can be used to create an installable software segment.

This document also provides some guidance on how to break up a software application and data into multiple segments (Sections 4.3 - 4.6). Refer to the DII COE Integration & Runtime Specification (I&RTS) document, Section 5.4 for additional guidance on determining the number and types of segments to break an application into.

1.3 Related Documentation

Although this document is intended to be as self contained as possible, developers are encouraged to be familiar with the contents of the complete set of DII COE documentation. Be sure to always reference the latest version for the documents listed below, and be aware that many of the documents are being modified to keep current with the current DII COE versions.

- DII COE Integration and Runtime Specification (I&RTS)
- DII COE Consolidated Version Description Document for HP
- DII COE Consolidated Version Description Document for Solaris
- DII COE Programming Guide for HP and Solaris
- DII COE System Administrators Guide for Windows NT, HP and Solaris
- DII COE API Reference Guide for Windows NT, HP and Solaris
- DII COE Installation Guide for HP
- DII COE Installation Guide for Solaris
- DII COE Installation Guide for Windows NT
- DII COE Programmers Reference Manual

- Installation Instructions, DII COE Developer's Toolkit
- DII COE User Profiles Application Programming Interface

2. Pre-Segmenting Activities

There are several activities that a developer must perform prior to starting to generate the segment descriptor files. They include ordering the COE, installing the DII COE and the Developer's Toolkit and identifying characteristics associated with the development environment of the application being segmented. The following paragraphs in this Chapter describe steps that should be taken before initiating the segmentation effort.

2.1 Order DII COE and DII COE Developer's Toolkit

Complete the following form and submit via email to gcssdctf@slidell.disa.mil for GCSS. GCCS must submit the form to their Service representative as identified in Table 2-1, who will then forward the request to DISA.

Table 2-1: GCCS Service Representatives

Service/Agency	Name	Telephone No.	DSN No.	Email Address
Army	D. Usechak	(908)427-3840	987-3840	usechak@doim6.monmouth.army
Navy	R. Walker	(703)602-2190	332-2190	walkerr@smtp-gw.spawar.navy.mil
Air Force	R. Paling	(617)271-4733	478-1186x4733	palingr@radium-vs1.hanscom.af.mil
Marine Corps	P. Sullivan	(619)725-9583	365-9583	paul.sullivan@mctssa-gw.usmc.mil
DMA	C. Blake	(703)275-8500		blakec@dma.gov
DODIIS ERB	S. Badger	(202)231-3534	428-3534	badgers@ix.netcom.net

NOTE: Most current POC's and product availability information is available on the DII COE homepage under Configuration Management (CM).

http://spider.osfl.disa.mil/dii/

NOTE: The most recent version of the DII COE baseline document will always have the latest on product and platform support. This can also be found on the DII COE homepage under CM.

SAMPLE REQUEST FOR COE VIA DCTF SLIDELL

SUBJECT: Request for DII COE Version 2.0
AUTHOR:
DATE:

TO: GCSS POC at DCTF Slidell or
Service Representative for GCCS

1. REASON FOR REQUEST: In order to support the migration of [Application] software to the DII COE, request 1 copy each of the annotated along with the appropriate documentation:

   - Windows NT 3.5.1 DII COE V2.0 Kernel
   - Windows NT 3.5.1 DII COE V2.0 Developer's Toolkit
   - Windows NT 3.5.1 DII COE Segment Templates
   - Sun Solaris 2.4 DII COE V2.0 Kernel
   - Sun Solaris 2.4 DII COE V2.0 Non-Kernel Components
   - Sun Solaris 2.4 DII COE V2.0 Developer's Toolkit
   - HP-UX 9.0.7 DII COE V2.0 Kernel
   - HP-UX 9.0.7 DII COE V2.0 Non-Kernel Components
   - HP-UX 9.0.7 DII COE V2.0 Developer's Toolkit

2. ADDITIONAL APPLICATION NAME(S):

   - Sybase 10.0.2 for HP
   - Sybase 10.0.2 for Sun
   - Oracle 7.2.2.4 for HP
   - Oracle 7.2.2.4 for Sun
   - Netscape News Server 1.1 for HP
   - Netscape News Server 1.1 for Sun
   - Netsite Server 1.1 for HP
   - Netsite Server 1.1 for Sun
   - Empire System Manager Agent 2.00b for HP
   - Empire System Manager Agent 1.353 for Sun
   - HP NetMetrix Remote Monitoring (RMON) Agent for HP
   - HP NetMetrix Remote Monitoring Agent for Sun
   - DCE Server 1.1 for HP
   - DCE Server 1.1 for Sun

3. PLATFORM: (Example PC, Client, or Server)

```
4.  OPERATING SYSTEM:

        •  Windows NT 3.51
        •  HP-UX 9.0.7
        •  Sun Solaris 2.4

5.  MEDIA:  (Example 3.5" diskettes)

        •  3.5" Floppies (Windows NT only)
        •  8mm Tape (SOL only)
        •  4mm Tape (HP only)

6.  GOVERNMENT POC: (Usually the D6 Service Representative)

7.  GOVERNMENT POC OFFICE SYMBOL:

8.  GOVERNMENT POC ADDRESS:

9.  GOVERNMENT POC PHONE:

10. PRIORITY:  (Example:  Mission Impact)

11. COMMENTS:  (Anything we need to know to process your segment.
```

Figure 2-1: DII COE Developer's ToolKit

2.2 Access to Text Editor

The procedures contained in this document do not address how files are created and/or edited. It is assumed that editors and/or word processors are available on the development platforms (i.e., vi and emacs for UNIX, WordPerfect and MS Word for NT, etc.). Files can be generated on any platform and transferred via file transfer protocol (FTP) to the UNIX platform.

2.3 Guidelines for Partitioning a Software Application into Segments

An application may be broken into two or more segments based on system architecture, maintainability and software architecture. Each segment is a separate, installable entity.

- **System Architecture** - If an application runs on several different machines and/or platforms simultaneously, each piece of this application should be a separate segment. For example, an application would have a segment for the database server and a segment for the client program. A different platform usually has a separate segment.
- **Maintainability** - If areas of the application have different life-cycle or maintainability requirements (e.g., data table that needs to be updated often or a program early in its development cycle that is changing rapidly), separate segments might be warranted.
- **Software Architecture** - Applications using imbedded COTS programs should break out the programs into its own segment for maintainability and so other applications can use the COTS segment (e.g., PERL, TCL/TK and WISH). If two separate applications share common code, this might call for another segment.

2.4 Guidelines for Creating Data Base System Files

TBD

2.5 Guidelines for Partitioning Data into Segments

TBD

2.6 Determine Segment Type

Determine the type of segment that is being created. Valid segment types are:

- **ACCOUNT GROUP**: A segment which serves as a template for establishing a runtime environment for individual operators.
- **COTS**: A segment totally comprised of vendor software and the product can be structured as a software segment outside of the mission application segment directory.
- **DATA**: A segment composed of a collection of data files for use by the system or by a collection of segments.
- **DATABASE**: A segment that is to be installed on a database server under the management of a DBMS and ownership of the Database Administrator. **This segment type is not supported in DII COE V2.0.**
- **SOFTWARE:** A collection of executables and static data items which extend base functionality and environment established by an account group.
- **PATCH**: A segment containing a correction to apply to another segment, whether data or software.

Refer to paragraph 5 of the DII COE I&RTS to determine which type of segment is appropriate. This information will be used in determining which segment descriptor files are required.

2.7 Determine Segment Attributes

Determine which of the following attributes apply to the segment:

- **AGGREGATE**: A collection of segments grouped together and managed as an indivisible unit.
- **CHILD**: A segment which is part of an aggregate, but is subordinate to a single mission application segment designated as the parent. An aggregate can have multiple child segments.
- **COE CHILD/COE PARENT**: A segment which implements functionality contained within the COE, as opposed to a mission application segment.
- **PARENT**: The mission application segment which is part of an aggregate that is considered to be the "root" segment. The parent segment name is the name presented to an operator as the name of the aggregate. An aggregate can have only one parent segment.

2.8 Determine Segment Name

Select a name for the segment consisting of a string of up to 32 alphanumeric characters (spaces permitted). The segment name will be used in several of the segment descriptor files.

2.9 Determine Segment Prefix

Select a prefix for the segment consisting of a string of up to 6 alphanumeric characters (no spaces). Although not required, the segment prefix should be named the same as the segment directory. It should be consistent with configuration management and segment registration process. The segment prefix selected cannot use reserved segment prefixes identified in paragraph 5.3 of the DII COE I&RTS.

> **NOTE:** DII COE NT V2.0 only supports 5 characters. This is an identified deficiency and will be fixed in a future release.

2.10 Identify Files, Paths and Directories

The developer should identify the files and directories required at run-time that make up a segment. The developer should also identify any specific location dependencies of those files. This list will come in handy when it is time to move segment files into required directories.

2.11 Identify Segment Dependencies

Dependencies on hardware, Commercial Off the Shelf (COTS) / Government Off the Shelf (GOTS) or other segments needed by the software module (or modules of an aggregate segment) should be documented for use during segment creation and testing. Dependencies should be grouped by type (e.g., other segments, operating system/APIs, data, and COTS products). Dependencies on segments provided by the DII COE Kernel do not need to be specified.

2.12 Identify Computing Resources

The developer should have the following resources identified.

2.12.1 Platform

Identify one of the supported DII COE platforms:

- **PC** - Defined for all 80x86 platforms regardless of OS
- **PC386** - Defined for INTEL PC386 workstations
- **PC486** - Defined for INTEL PC486 workstations
- **PENTIUM** - Defined for INTEL PENTIUM workstations

- **SPARC** - Defined for Sun Sparc workstations
- **SUN4** - Defined for Sun 4 workstations
- **SOL** - Defined for Sun Sparc workstations running Solaris
- **SUN** - Defined for Sun 4 workstations running SunOS

- **HP** - Defined for HP platforms running HP-UX
- **HP700** - Defined for HP700 series workstations
- **HP712** - Defined for HP712 workstations

- **HP715** - Defined for HP715 workstations
- **HP755** - Defined for HP755 workstations

2.12.2 Size

Identify size in kilobytes of memory and disk spaces:

- **Disk** - is the size of the segment, including all of its subdirectories, at install time;
- **Memory** - is the size of RAM required;
- **Partition** - is used to identify disk size needed to reserve space on multiple disk partitions;
- **Temporary Space** - used to allocate temporary disk space. The amount of temporary space will be factored in when determining if there is sufficient disk space to install the segment by the COE Installer.

NOTE: Remote disks used in the DII COE directory structure should be labeled "home1, home2…homen". The COE installer software will automatically mount these disks as required.

2.12.3 Operating System

Identify one of the supported DII COE operating systems:

- HP-UX
- NT
- SOL

2.13 Identify Icons

The developer should identify the icon files that are to be made available on the desktop to launch segment functions. See paragraph 5.5.14 of the DII COE I&RTS for information on icons.

2.14 Identify File Menus

The developer should identify the menu files contained within the segment. See paragraph 5.5.16 of the DII COE I&RTS for information on Menus.

2.15 Identify FONTS

The developer should identify any font files, not already supported by the DII COE, required for the application. The current font files support are as follows:

2.16 Register Segments

Segment registration is the beginning process into the DII COE. It's purpose is to collect information about the segment for publication in a *segment catalog*. This "catalog" will be maintained in a software repository and will simply know what capabilities exist within the DII COE. The segment catalog will be made available on-line through an HTML browser and will contain the information provided by

developers in a segment registration form. Keyword searches can be performed on the segment catalog by developers to identify the potential for reusable segments, or by operational sites to find new mission applications.

Currently there are two processes for registering segments. One is for GCCS which is forwarded to the DISA's Operational Support Facility (OSF), Configuration Management and the other is for GCSS which is forwarded to DISA Continuity of Operations and Test Facility (DCTF), GCSS Operations Center.

2.16.1 GCCS Segment Registration

The GCCS segment registration form includes the following information:

- segment name
- segment prefix
- segment directory name
- list of related segments
- program management point of contact
- technical point of contact
- process point of contact
- estimated memory required by the segment
- estimated disk storage requirements
- platform availability (PC only, Solaris only, etc.)
- short paragraph describing the segment features
- list of keywords for use in catalog searches
- unclassified picture of the segment user interface (GIF, JPEG, or X11 Bitmap format).

Not all information provided at segment registration time is made available to the community at large. The *technical point of contact* is available only to the DISA Engineering Office in the event that technical questions or issues arise during segment integration. The *process point of contact* is the individual authorized by the segment program manager to actually submit the segment, or to receive status information and notifications. The *program management point of contact* is the only individual authorized to commit schedule or resources, and is the only individual authorized to release information about the segment to the community at large. The three points of contact are selected by the service/agency responsible for the segment. Services may elect to designate a single individual for all three points of contact, and may include an alternate point of contact for each category.

2.16.2 GCSS Segment Registration

A completed GCSS Segment Registration form must be submitted to the DCTF by the segment sponsor/developer as early in segment development as possible. Registration shall be done before the segment can be accepted by the DCTF for GCSS certification evaluation, and should be done near the end of software migration/development planning but before any migration or development work is begun (if possible).

Attached as Appendix A is the DCTF GCSS Segment Registration and Delivery Package with enclosures.

3. New Segment Development - UNIX Environment

The procedures in this section can be used to develop new segments from newly developed and/or migrated software applications that are intended to execute in a UNIX runtime environment. These procedures assume a basic understanding of UNIX commands and the UNIX runtime environment. The process of segmenting an application is shown in Figure 3-1.

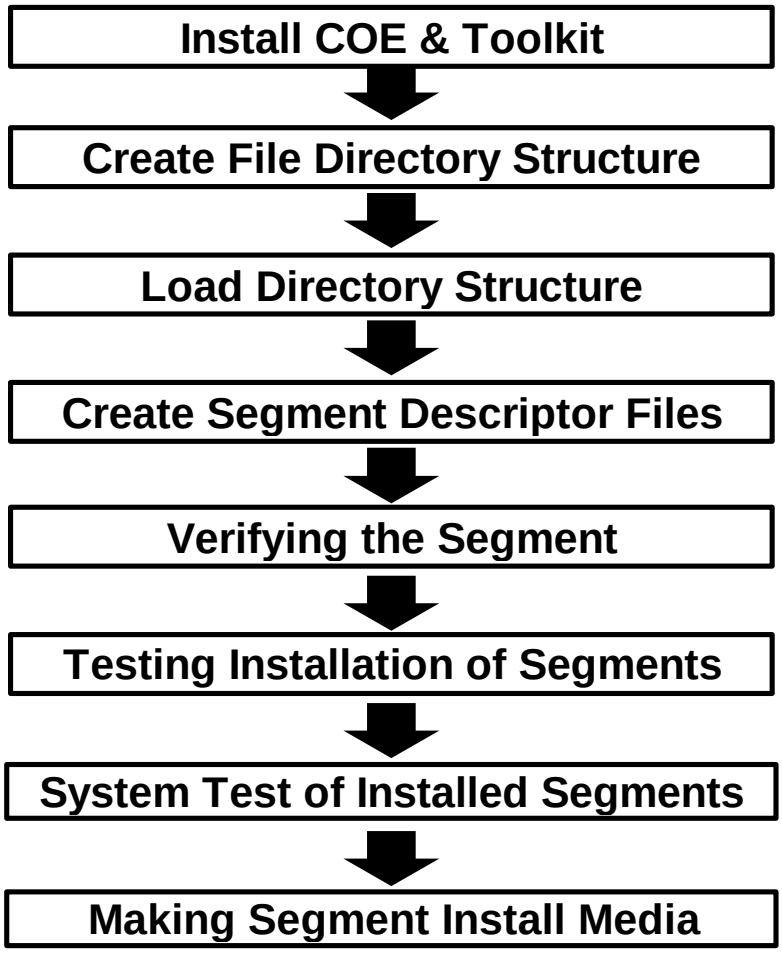

Figure 3-1: DII COE Segmentation Process

3.1 Installing the DII COE Unix Kernel and Developer's Toolkit

The developer has two options prior to segmenting in establishing an environment for segmentation. The two options are:

- Load the DII COE Kernel and the Developer's Toolkit.

  OR

- Load only the Developer's Toolkit. The SegName segment descriptor file for the SysAdm account group must be created by the developer in order to successfully execute VerifySeg on the newly created segment. The directory for the SegName file can be generated by executing the following commands:

  >> *mkdir -r /h/AcctGrps/SysAdm/SegDescrip*

  >> *cd /h/AcctGrps/SysAdm/SegDescrip*

  Generate the following SegName file:

```
$TYPE:ACCOUNT GROUP
$NAME:System Administrator
$PREFIX:SysAdm
```

  Skip to paragraph 3.1.3.

3.1.1 Installing the DII COE Unix Kernel

The developer must load the DII COE Kernel onto the development platform in order to have the ability to complete the development effort and to test the application software (once segmented) within the DII COE runtime environment.

> **NOTE**: There should be at least 150 megabytes identified for the "/h" directory on the root disk. The DII COE Kernel must be loaded on the root disk and not on the remote disk.

The DII COE is loaded in three parts:

- Bootstrap DII COE
- Remaining DII COE Kernel
- Remaining DII COE (other DII COE segments, several tapes)

The Bootstrap DII COE consists of the operating system, windowing environment, operating system and windowing patches, Security Administration software, and System Administration software.

> **NOTE**: Developers will be required to provide purchase order information to distributors for authentication prior to the release of any DII COE COTS segments.

DII COE segments that require vendor authorization to ensure valid licensing agreements are shown in Table 3.1. All of the software items requiring license by the developer are in the DII COE Infrastructure and Support Applications Services, not the Kernel Tape. The developer should be able to complete the segmentation process with just the COE Kernel software, thus the software licenses may not be required. However, if Oracle or one of the other products listed in Table 3-1 is required then the developer will be required to show licenses.

Table 3-1: DII COE License Requirements

Sun	HP
Oracle 7.2.2.4 or Sybase 10.0.02	HP-UX 9.07
DCE Server 1.1	Oracle 7.2.2.4 or Sybase 10.0.02
EMPIRE 1.353	DCE Server 1.1
NetMetrix 4.5	EMPIRE 2.00b
Netsite Server 1.1	NetMetrix 4.50
Netscape News 1.1	Netsite Server 1.1
	Netscape News
	STREAMS 1.0

Installation of the DII COE Kernel software is accomplished by the developer following the step by step procedures in the DII COE Kernel Installation Guide (specific to HP or Sun). There are several tapes associated with the install. It is up to the developer to determine what DII COE segments are required. Table 3-2 contains the list of tapes for the 2.0 release.

Table 3-2: DII COE Release 2.0 Tapes

Sun	HP
DII COE Kernel Tape	DII COE Kernel Tape
DII COE Consolidated Developer's ToolKit Tape	DII COE Consolidated Developer's ToolKit Tape
DII COE Applications Tape	DII COE Applications Tape
DII COE Oracle Segment Tape	DII COE Oracle Segment Tape
DII COE Sybase Segment Tape	DII COE Sybase Segment Tape
DII COE DCE Server Segment Tape	DII COE DCE Server Segment Tape
DII COE EMPIRE Segment Tape	DII COE EMPIRE Segment Tape
DII COE NetMetrix Segment Tape	DII COE NetMetrix Segment Tape
DII COE Netsite Server Segment Tape	DII COE Netsite Server Segment Tape
DII COE Netscape News Segment Tape	DII COE Netscape News Segment Tape
	DII COE STREAMS SEGMENT TAPE

3.1.2 Verifying the UNIX DII COE Successfully Loaded

Verify that the DII COE loaded by rebooting the system using the following steps:

Step 1: Login as sysadmin.

Step 2: Select the reboot option from the System pull-down menu.

Step 3: Confirm that the system reboots and prompts the user to login.

> **NOTE**: Once the DII COE has been loaded and the system re-booted, all access will be controlled by the COE Common Desktop Environment (CDE). If access to a UNIX command line is required, the operator will have to log in as "sysadmin", invoke the Application Manager by clicking on the "File Cabinet Drawer", and select the DII Icon. This will give the operator access to both an "Xterm" and "Dterm" Icon. Accessing either of these two icons will provide a UNIX prompt.

3.1.3 Installing DII COE Unix Developer's Toolkit

The tools required to support development and runtime capabilities make-up the DII COE Developer's Toolkit. Specifically, the DII COE Toolkit contains the following:

- API Libraries and Object Code
  - Archive: libCOETools.a
  - Archive: libaudit.a
  - Archive: libdce.a
  - Several JMTK libraries
- C Header Files for Development Tools APIs
  - Header File: DIICOETools.h
- C Header Files for Print Services APIs
  - Header File: PrintAPI.h
  - Header File: PrintSPI.h
- C Header Files for DCE APIs
  - Several DCE header files
- C Header Files for JMTK APIs
  - Several JMTK header files
- On-Line UNIX Man Pages for APIs
- COE Development Tools (See I&RTS Appendix C)
  - Executable File: CalcSpace
  - Executable File: CanInstall
  - Executable File: ConvertSeg
  - Executable File: MakeAttribs
  - Executable File: MakeInstall
  - Executable File: TestInstall
  - Executable File: TestRemove
  - Executable File: TimeStamp
  - Executable File: VerifySeg
  - Executable File: VerUpdate

- Examples for using the DII COE Runtime Tools and Print Services.

> **NOTE**: All entries to be typed at the UNIX command prompt or put into a UNIX file are shown using *italics font* throughout this document. Items which are underlined require the developer to substitute either segment specific information or development environment specific information.

The steps to install the DII COE Developer's Toolkit are as follows:

Step 1: Load the tape onto the tape drive of the development platform.

Step 2: Login as root.

> **NOTE:** Loading the Developer's Toolkit while logged in as root is recommended by DII COE documentation. However, the Toolkit can be loaded while logged in as any user and the appropriate changes to the file and directory permissions can be made to provide access to the appropriate users, groups, etc. Loading the Toolkit as a regular user does require the System Administrator to create the /h directory and allow write access to the appropriate user or group(s).

Step 3: Type the following command to get into a cshell environment:

>> *csh*

> **NOTE**: Loading the Developer's Toolkit and DII COE assumes that the runtime environment is setup in a cshell environment. If you wish to use another shell, other script files will have to be modified instead of the .cshrc script file as noted below.

Step 4: Make an "/h" directory if one does not already exist by typing:

>> *mkdir /h*

> **NOTE:** COE maybe installed anywhere. It is recommended that it be installed under /h for consistency.

Step 5: Execute the following commands to install the Toolkit:

>> cd /h
>> tar xvf /dev/rmt/3mn

substituting your tape drive name for "/dev/rmt/3mn"

> **NOTE**: The extraction process will build "/h/TOOLS/DII_DEV", "/h/TOOLS/JMTK" and "/h/TOOLS/DCE" directories. Subdirectories under each will also be dynamically established.

Step 6: Add the COE development tool directory to the UNIX path environment variable by editing the .cshrc file and adding "/h/TOOLS/DII_DEV/bin" to the pathnames list for all user accounts that will need to execute the DII COE Developer's tools.

Step 7: Add "/h/TOOLS/DII_DEV/man" to the search path for UNIX man pages in the .login file for any user accounts that will need to execute the DII COE Developer's tools.

Step 8: Edit the .cshrc file for each user that will be accessing the DII COE Developer's tools and add the following line:

source /h/TOOLS/DII_DEV/Scripts/MakeTOOLSEnv

This will establish the MACHINE, MACHINE_CPU, MACHINE_OS, and TOOLS_HOME environment variables.

Step 9: Respond to the "Enter the tools home dir>" (default is /h/TOOLS/DII_DEV) prompt by typing:

/h/TOOLS/DII_DEV/

The directory structure created by extracting the Developer's Toolkit is shown in Figure 3-2. The archive needed to link developer tools to the COE development tools is the "libCOETools.a" file located in the "/h/TOOLS/DII_DEV/lib" subdirectory. The executables for the tools that can be run from a command line are contained in the "/h/TOOLS/DII_DEV/bin" subdirectory. The header file needed to access APIs for these tools is contained in "/h/TOOLS/DII_DEV/include/DIICOETools.h". The man pages for the COE tool APIs are located in the "/h/TOOLS/DII_DEV/man" subdirectory.

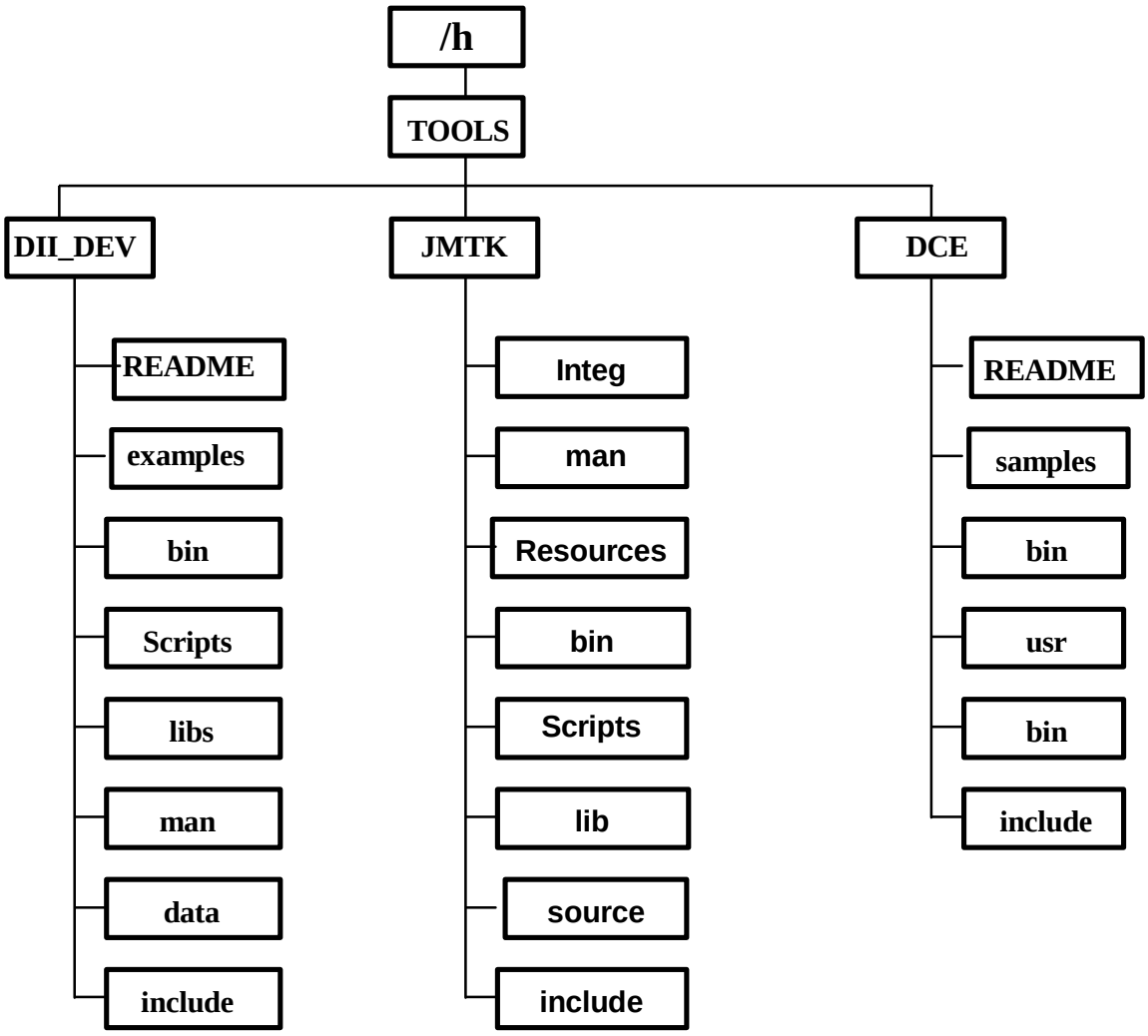

Figure 3-2: DII COE Unix Developer's Toolkit Directory Structure

3.2 Create Unix File Directory Structure

This document describes the establishment of a development environment by creating a directory structure and moving files to specific subdirectories. It provides the necessary environment to build the required segment descriptor files, script files, man pages, icons, menus, fonts, application default files, header files, data files, library files, and database management system (DBMS) files required for the segment. There are mandatory and optional subdirectories in the directory structure.

The directory structure is similar to the runtime directory structure described in Chapter 5 of the DII I&RTS. The TOOLS subdirectory will have been created as a result of installing the Toolkit as described in section 4.1 Install DII COE Developer's Toolkit.

Create a segment development directory structure as shown in Figure 3-3 using the following steps:

Step 1: Change umask to set default read/write/execute permissions on all created files and directories to support segmentation by typing the following command:

>> umask 027

to give read/write/execute for the user and read/execute for other users within the same group. See a UNIX manual if a different setting is desired.

NOTE: Ensure privileges are right before running VerifySeg.

Step 2: A subdirectory for any and all segments being developed is required under "/Dev". A segment prefix must be defined for each segment. The segment prefix is used as the subdirectory name and will be used later in many of the segment descriptor files. Use the following commands to create the segment directory:

>> mkdir /Dev
>> cd /Dev
>> mkdir SegmentPrefix

Step 3: The runtime environment requires "SegDescrip", "Scripts", "bin", "data", "man", "include", "lib", and "Integ" subdirectories under each segment if needed. Use the following commands to create these directories:

>> cd SegmentPrefix
>> mkdir SegDescrip Scripts bin data man lib Integ

Step 4: Create the additional subdirectories using the following commands:

>> cd /Dev/SegmentPrefix/data
>> mkdir icons menus fonts app-defaults

Step 5: If a database type segment is being developed, generate additional directories using the following commands:

>> cd /Dev/SegmentPrefix
>> mkdir DBS_files Install

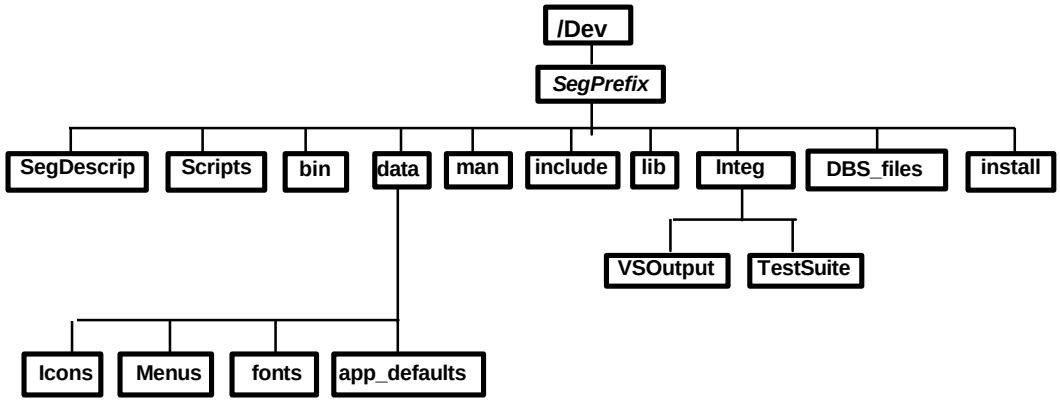

Figure 3-3: Unix Segment Development Directory Structure

3.3 Load Unix Directories with Appropriate Files

Preparing a segment for submission to DISA requires the appropriate files for compiling, linking and loading the application. Perform the following steps to load the COE development environment with the required files (Note: *SegmentPrefix* is the prefix assigned to the segment by the developer).

3.3.1 Executables Files

Copy executable files associated with the segment into "/Dev/*SegmentPrefix*/bin".

3.3.2 Header Files

Copy header files for public APIs associated with the segment into "/Dev/*SegmentPrefix*/include". Public header files are those header files that contain APIs that other segments have access to.

3.3.3 Man Page Files

Man pages are required for submitting a segment to DISA. Copy UNIX Man pages associated with the segment into "/Dev/*SegmentPrefix*/man".

3.3.4 Object Code Library Files

Copy public object code libraries (i.e. API's) associated with the segment into "/Dev/*SegmentPrefix*/lib". These files are not used at runtime but are provided to be available to developers through the future releases of the DII COE Developers Toolkit..

3.3.5 Data Files

Copy data files associated with the segment into "/Dev/*SegmentPrefix*/data". Data files may be in a directory structure under this directory.

3.4 Create Required Segment Descriptor Files

This section describes the procedures for generating the **required** segment descriptor files for all segment types. The files include ReleaseNotes, SegInfo, SegName, and VERSION.

Step 1: Execute the following command to get into the correct directory for creating descriptor files.

>> chdir /Dev/SegmentPrefix/SegDescrip or
>> cd /Dev/SegmentPrefix/SegDescrip

Step 2: Create the following descriptor files using an editor or by using the appropriate tools as specified.

3.4.1 SegName

The SegName segment descriptor file is required for all segment types. The SegName segment descriptor file contains information identifying the segment being created. Most of this information was determined back in paragraphs 2.6, 2.7, 2.8 and 2.9 of this document.

Step 1: Generate the SegName segment descriptor file using the following format:

$TYPE[1]:segment type[2][:attribute[3]]
$NAME[4]:name[5]
$PREFIX[6]:SegmentPrefix[7]

$SEGMENT[8]*:name*[5]*:SegmentPrefix*[7]*:home dir*[9] ***OR***
$CHILD[10]*:name*[5]*:SegmentPrefix*[7]*:home dir*[9] ***OR***
$PARENT[11]*:name*[5]*:SegmentPrefix*[7]*:home dir*[9]

[1] **$*TYPE***: DII COE Keyword used to set segment type. The $TYPE keyword must be defined for all segment types.

[2] ***segment type***: COTS, ACCOUNT GROUP, SOFTWARE, DATA, or PATCH.

The **COTS** segment type is used if the segment is vendor provided.

The **ACCOUNT GROUP** segment type is used if account specific icons, menus, and script files are being developed for a non-COE standard account group.

The **SOFTWARE** segment type is used for mission application software or COE application software that is custom developed.

The **DATA** segment type is used if only local and global data files are being created/submitted as a segment.

The **PATCH** segment type is used to create replacement files for segments that have already been delivered. Only those files that have been revised are submitted as a PATCH segment.

[3] ***attribute***: AGGREGATE , CHILD, COE CHILD, COE PARENT. The AGGREGATE attribute is indicated if the segment being created is the parent segment of a group of mission application segments. The CHILD attribute is used if the segment is a child of a parent mission application segment. The COE CHILD attribute is used if the segment is a child of the parent COE segment. The COE PARENT attribute is used if the segment is to be the parent COE segment in a COE variant.

[4] ***$NAME***: DII COE Keyword to set the segment name. The $NAME keyword must be defined for all segment types.

[5] ***name***: segment name consisting of a string of up to 32 alphanumeric characters (spaces permitted).

[6] ***$PREFIX***: DII COE Keyword to set the segment prefix. The $PREFIX keyword must be defined for all segment types.

[7] ***SegmentPrefix***: segment prefix consisting of a string of up to 6 alphanumeric characters (no spaces, used in directory structure). Should be consistent with configuration management and segment registration process. Cannot use reserved segment prefixes identified in paragraph 5.3 of the DII COE I&RTS.

[8] ***$SEGMENT***: DII COE Keyword in the file is optional and used to list the affected segments or account groups that a patch segment applies to. This keyword is required if neither $CHILD or $PARENT are used. Multiple affected segments or child segments may be listed by listing each segment on a separate line.

[9] ***home dir***: /h/*SegmentPrefix*

[10] ***$CHILD***: DII COE Keyword is optional and used to list the children if and only if the $AGGREGATE or $COE PARENT attributes have been listed. Multiple $CHILD lines can be used in this file.

[11] ***$PARENT:*** DII COE keyword is optional and used to list the parent if and only if the $CHILD or $COE CHILD attributes have been listed. Only one line with $PARENT can be included in the file.

Step 2: Review the file on the screen and verify that the correct segment identification information is contained in the file using the following command:

>> cat SegName

Step 3: Move the file to the "/h/SegPrefix/SegDescrip" directory (if necessary).

Refer to paragraph 5.5.31 of the DII COE I&RTS for more detailed information.

3.4.2 VERSION

The VERSION segment descriptor file is required for all segment types. The VERSION segment descriptor file contains the time and date of when the segment was created. It must be created using an editor and then updated by using the **TimeStamp** tool.

Step 1: Generate the Version segment descriptor file using the following format:

version #[1]:date[2]

[1] ***version #*** = developer specified version number of the segment using the form a.b.c.d where a = major release; b = minor release; c = maintenance release, d = developer release. Refer to I&RTS paragraph 3.1 for additional information on version numbering.

[2] ***date*** = date that the segment was created (or last modified) using the form mm/dd/yy.

Step 2: Move the file to the "/h/SegPrefix/SegDescrip" directory (if necessary).

Step 3: Execute the TimeStamp tool to automatically update the Version descriptor file with the current time and date using the following command:

>> TimeStamp

Step 4: Review the file on the screen and verify that the current date is contained in the file using the following command:

>> cat VERSION

Refer to paragraph 5.5.34 of the DII COE I&RTS for more detailed information.

3.4.3 Release Notes

The ReleaseNotes segment descriptor file is required for all segment types. The ReleaseNotes descriptor file contains information that identifies the new or enhanced functionality being provided by the segment.

Step 1: Generate the ReleaseNotes file and include information of interest to an operator. **Do not** include information on version, point of contacts, phone numbers or help information. **Do** include known problems that have been fixed and new features introduced by this release. This file cannot include tabs or embedded control characters.

Step 2: Move the file to the "/h/*SegmentPrefix*/SegDescrip" directory (if necessary).

Refer to paragraph 5.5.25 of the DII COE I&RTS for more detailed information.

3.4.4 SegInfo

The SegInfo segment descriptor file is required for all segment types. The SegInfo file contains several types of information which is used to integrate and install the segment. The SegInfo sections required depend upon the type of segment being created. Table 3-3 shows the required and optional SegInfo sections for each segment type.

The Hardware and Security sections of SegInfo are required for all segment types. Some of the remaining SegInfo sections described below are required for some segment types and some SegInfo sections are optional for some segment types. Each SegInfo section includes a section heading in square brackets followed by keywords, commands, filenames, directory names, pathnames, etc.

Table 3-3: SegInfo Section Dependencies per Segment Type

SegInfo Header	Software Segment	Account Group Segment	COTS Segment	Data Segment	Patch Segment
AcctGroup	**N**	**R**	**N**	**N**	**N**
COEServices	**O**	**O**	**O**	**O**	**O**
Community	**O**	**O**	**O**	**O**	**O**
Comm.deinstall	**O**	**O**	**O**	**O**	**O**
Compat	**O**	**O**	**O**	**O**	**O**
Conflicts	**O**	**O**	**O**	**O**	**O**
Data	**N**	**N**	**N**	**R**	**N**
Database	**Reserved**	**Reserved**	**Reserved**	**Reserved**	**Reserved**
Direct	**O**	**O**	**O**	**O**	**O**
FilesList	**O**	**O**	**R**	**O**	**O**
Hardware	**R**	**R**	**R**	**R**	**R**
Icons	**O**	**R**	**O**	**N**	**O**
Menus	**O**	**R**	**O**	**N**	**O**
Network	**N**	**N**	**N**	**N**	**O**
Permissions	**O**	**O**	**N**	**N**	**O**
Processes	**O**	**O**	**O**	**N**	**O**
ReqrdScripts	**O**	**R**	**N**	**N**	**N**
Requires	**O**	**O**	**O**	**O**	**O**
Security	**R**	**R**	**R**	**R**	**R**

R = Required, O = Optional, N = Not Required

3.4.4.1 Hardware

The Hardware section of SegInfo is required for all segment types. It specifies the computing resources required by the segment. Specifically, it identifies the platform using a predefined keyword, the disk space in kilobytes, the amount of disk space and growth disk space in kilobytes, the disk partitions and growth disk partitions required in kilobytes, the operating system using a predefined keyword, and the amount of temporary space required during installation.

Step 1: Generate the Hardware section of SegInfo using the following format:

[Hardware]
$CPU[1]:platform[2]
$MEMORY[3]:size[4]
$DISK[5]:size[6][:reserve][7] ***or*** *$PARTITION[8]:diskname[9]:size[10][:reserve[7]]*
•

•
•

$PARTITION[8]*:diskname*[9]*:size*[10]*[:reserve*[7]*]*
$OPSYS[11]*:operating system*[12]
$TEMPSPACE[13]*:size*[14]

[1] ***$CPU*** = DII COE Keyword used to establish platform type. The $CPU keyword must be defined for all segment types.

[2] ***platform*** = target runtime platform dependency and is identified using one of the following:

ALL: Platform Independent

PC Defined for all 80x86 platforms regardless of OS

PC386 Defined for INTEL PC386 workstations

PC486 Defined for INTEL PC486 workstations

PENTIUM Defined for INTEL PENTIUM workstations

SPARC Defined for Sun Sparc workstations

SUN4 Defined for Sun 4 workstations

SOL Defined for Sun Sparc workstations running Solaris

SUN Defined for Sun 4 workstations running SunOS

HP Defined for HP platforms running HP-UX

HP700 Defined for HP700 series workstations

HP712 Defined for HP712 workstations

HP715 Defined for HP715 workstations

HP755 Defined for HP755 workstations

[3] ***$MEMORY*** = DII COE Keyword used to define segment memory requirements. The $MEMORY keyword is required for all segments except for DATA segments.

[4] ***size*** **(MEMORY)** = amount of RAM required by the segment in Kilobytes

[5] ***$DISK*** = DII COE Keyword used to define segment disk requirements.

[6] ***size*** **(DISK)** = size of the segment (and all subdirectories) at install time expressed in Kilobytes. Once this value is established, the COE tool CalcSpace can be used to automatically calculate the size of the segment and update the $DISK keyword accordingly.

[7] ***reserve*** = amount of extra RAM in Kilobytes reserved to accommodate future growth of the segment.

[8] ***$PARTITION*** = DII COE Keyword used to define segment partitions.

[9] ***diskname*** = explicit partition name (e.g., /home2) or an environment variable name of the form DISK1, DISK2, ...DISK99. The installation software will set the environment variables DISK1, DISK2, etc... to the absolute pathname for where space has been allocated. $Partition keywords are assumed to be in sequential order.

[10] ***size*** **(PARTITION)** = size of the segment in Kilobytes on a particular disk partition. Installation software does not allow a segment to be split across multiple disk partitions. However, the PostInstall script does allow for splitting the segment across multiple disk partitions.

[11] ***$OPSYS*** = DII COE Keyword used to define operating system requirement of the segment.

[12] ***operating system*** = one of the DII COE supported operating systems listed in I&RTS paragraph 5.3 for MACHINE_CPU. They include HPUX, NT, SCO, SOL, SUNOS, WIN31, and WIN95.

[13] ***$TEMPSPACE*** = DII COE Keyword used to define Temporary space requirements of the segment.

[14] ***size*** **(TEMPSPACE)** = amount of temporary diskspace in kilobytes that are used during the installation process. If space is available, the installation software sets the variable COE_TMPSPACE to the absolute pathname where space is allocated. If not enough space can be found, an error message is displayed to the operator.

Refer to paragraph 5.5.13 of the DII COE I&RTS for more detailed information.

3.4.4.2 Security

The Security section of the SegInfo segment descriptor file is required for all segment types. The Security section of the SegInfo segment descriptor file indicates the highest classification level for the segment.

Step 1: Generate the Security section of the SegInfo descriptor file using the following format:

```
[Security]
classification level[1]
```

[1] ***classification level*** = UNCLASS, CONFIDENTIAL, SECRET, or TOP SECRET indicating the segment classification level of the segment.

Refer to paragraph 5.5.28 of the DII COE I&RTS for more detailed information.

3.4.4.3 AcctGroup

The AcctGroup section of the SegInfo descriptor file is required for Account Group segments and optional for Aggregate Segments. The entry of the file is used as a template for groups within the UNIX "/etc/passwd" file.

Step 1: Generate the AcctGroup section of the SegInfo segment descriptor file using the following format:

```
[AcctGroup]
$group name[1]:group ID[2]:shell[3]:profile flag[4]:home dir[5]:default profile name[6]
$CLASSIF:classification[7]
```

[1] ***group name*** = Alphanumeric string up to 15 characters used to identify this account group. The account group name must be unique (i.e., no other account group may have the same name).

[2] ***group id*** = UNIX group ID to be inserted into the password file for accounts created from this group.

[3] ***shell*** = UNIX shell to execute when logging in (e.g., */bin/csh*, */bin/sh*)

[4] ***profile flag*** = 0 if no profiles are allowed, otherwise 1.

[5] ***home dir*** = Home directory for the account group (e.g., */h/AcctGrps/SecAdm*). The path name may contain up to 256 characters.

[6] ***default profile name*** = Alphanumeric string up to 15 characters identifying the account group's default profile. This name is ignored unless the profile flag is non-zero.

[7] ***Classification level*** = Classification of the profile and can be UNCLASS, CONFIDENTIAL, SECRET, or TOP SECRET. The classification level for a profile will default to TOP SECRET unless the segment has defined it otherwise.

Refer to paragraph 5.5.1 of the DII COE I&RTS for more detailed information.

3.4.4.4 COEServices

The COEServices section of the SegInfo segment descriptor file is used to specify changes in services provided by the operating system. The COEServices section of the SegInfo segment descriptor file is optional for all segment types.

Step 1: Generate the COEServices section of the SegInfo segment descriptor file using the following format:

```
[COEServices]
$GROUPS
groupname[1]:group id[2]
groupname[1]:group id[2]
        •
        •
groupname[1]: group id[2]
$PASSWORDS
login name[3]:user id[4]:group id[2]:comment[5]:home dir[6]:shell[7]
login name[3]:user id[4]:group id[2]:comment[5]:home dir[6]:shell[7]
        •
        •
login name[3]:user id[4]:group id[2]:comment[5]:home dir[6]:shell[7]

$SERVICES[:servicescomment[11]]
socketname[8]:port[9]:protocol[10]{:alias[12]}
socketname[8]:port[9]:protocol[10]{:alias[12]}
        •
        •
socketname[8]:port[9]:protocol[10]{:alias[12]}
```

[1] ***groupname*** = corresponds to the groupname used in the UNIX *"/etc/group file"*.

[2] ***group id*** = corresponds to the group id used in the UNIX *"/etc/group file"*.

[3] ***login name*** = corresponds to login name field used in the UNIX *"/etc/passwd file"*.

[4] ***user id*** = corresponds to the user id field used in the UNIX *"/etc/passwd file"*.

[5] ***comment*** = corresponds to the comment field used in the UNIX *"/etc/passwd file"*.

[6] ***home dir*** = corresponds to the home dir field used in the UNIX *"/etc/passwd" file*.

[7] ***shell*** = corresponds to the shell field used in the UNIX *"/etc/passwd" file*.

[8] ***socketname*** = name of the socket to be added to the *"/etc/services"* system file.

[9] ***port*** = port number requested. Numbers 2000-2999 are reserved for COE segments.

[10] ***protocol*** = *tcp* or *udp*.

[11] ***servicescomment*** = comment to be included in the *"/etc/services"* system file for the port.

[12] ***alias*** = symbolic name referring to the assigned port.

Requests to modify the "/etc/services" file which adds sockets is done through the COEServices section of the SegInfo segment descriptor file. This control point for requests to add socket names and ports helps avoid conflicts between segments. Port numbers in the range 2000-2999 are reserved for COE segments (Segment has a COE PARENT or COE CHILD attribute specified in the SegName segment descriptor file). Segments should avoid creating sockets with port numbers less than 1000 since these are generally reserved for operating system usage.

Refer to paragraph 3.4.2.2, COEServices, of this document if a port is required to be added by the operating system for the application being segmented.

Refer to paragraph 5.5.2 of the DII COE I&RTS for more detailed information.

3.4.4.5 Community

The Community section of the SegInfo segment descriptor file is used to direct the installation of software for which no corresponding segment descriptor file exists. It acts as a "catch-all" to insert, delete, append and replace blocks of text in ASCII files. The Community section of the SegInfo segment descriptor file is optional for all segment types.

Step 1: Generate the "Community" section of the SegInfo segment descriptor file using the following format:

```
[Community]
$FILE filename[1]
$APPEND
{
stuff to be appended to filename
}
$FILE filename[1]
$COMMENT:char[2]
{
comment out the this text using the char character in file filename.
}
$FILE filename[1]
$DELETE
{
delete this block of text from file filename.
}
$FILE filename[1]
$INSERT
{
Find the first occurrence of this text in file filename and insert the text that follows.
```

```
}
{
Insert this block of text in file filename.
}
$FILE filename[1]
$REPLACE
{
Find the first occurrence of this text in file filename and replace it with the text that
follows.
}
{
Replace with this block of text in file filename.
}
$FILE filename[1]
$SUBSTR: DELETE [ALL] | INSERT [ALL] | REPLACE [ALL]
{
text string to be deleted, replaced or location of additional text to be inserted
}
{
text string to be inserted into or replaced with.
}
$FILE filename[1]
$UNCOMMENT: char[3]
{
text to be uncommented using character char.
}
```

[1] ***filename*** = file that commands are to be acted on.

[2] ***char*** (COMMENT) = character to comment out text strings/blocks.

[3] ***char*** (UNCOMMENT) = character to delete for uncommenting text strings/blocks.

Refer to paragraph 5.5.3 of the DII COE I&RTS for more detailed information.

3.4.4.6 Comm.deinstall

The Comm.deinstall section of the SegInfo segment descriptor file is used to undo what was done by the community segment descriptor section of SegInfo. It acts as a "catch-all" to insert, delete, append and replace blocks of text in ASCII files to return them to their original state. The Comm.deinstall section of the SegInfo segment descriptor file is optional for all segment types.

Step 1: Generate the "Community" section of the SegInfo segment descriptor file using the following format:

```
[Comm.deinstall]
$FILE filename[1]
$APPEND[2]
{
stuff to be appended to filename
}
$FILE filename[1]
$COMMENT[3]:char[4]
{
```

comment out the this text using the char character in file filename.
}
$FILE filename[1]
$DELETE[5]
{
delete this block of text from file filename.
}
$FILE filename[1]
$INSERT[6]
{
Find the first occurrence of this text in file filename and insert the text that follows.
}
{
Insert this block of text in file filename.
}
$FILE filename[1]
$REPLACE[7]
{
Find the first occurrence of this text in file filename and replace it with the text that follows.
}
{
Replace with this block of text in file filename.
}
$FILE filename[1]
$SUBSTR[8]*: DELETE [ALL] | INSERT [ALL] | REPLACE [ALL]*
{
text string to be deleted, replaced or location of additional text to be inserted
}
{
text string to be inserted into or replaced with.
}
$FILE filename[1]
$UNCOMMENT: char
{
text to be uncommented using character char.
}

[1] ***filename*** = filename of which the following commands apply to.

[2] **$APPEND** = DII COE Keyword to undo a Community DELETE command at the end of file filename.

[3] **$COMMENT** = DII COE Keyword to undo a Community UNCOMMENT in file filename.

[4] ***char*** **(COMMENT)** = character to comment out text strings/blocks.

[5] The **$DELETE** = DII COE Keyword command to undo a Community APPEND or INSERT command in file filename.

[6] The **$INSERT** = DII COE Keyword command to undo a Community DELETE command in file filename.

[7] The **$REPLACE** = DII COE Keyword command to undo a Community REPLACE command in file filename.

[8] The **$SUBSTR** = DII COE Keyword command to undo a Community SUBSTR command in file filename.

Refer to paragraph 5.5.4 of the DII COE I&RTS for more detailed information.

3.4.4.7 Compat

The Compat section of the *SegInfo* segment descriptor file is used to indicate the degree to which backward compatibility is preserved with the newly released segment. The Compat section of the *SegInfo* segment descriptor file is optional for all segment types.

Step 1: For segments which are backwards compatible with all previous releases of that segment, generate the Compat section of *SegInfo* by using the following format:

```
[Compat]
+ALL
```

For segments which are not backwards compatible with any previous releases of that segment, generate the Compat section of *SegInfo* by using the following format:

```
[Compat]
-NONE
```

For segments which are backwards compatible with specific previous releases of that segment starting from an earliest release, generate the Compat section of *SegInfo* by using the following format:

```
[Compat]
$EARLIEST
version[1]
```

[1] ***version*** = specific version number of the DII COE version format "a.b.c.d".

For segments which are backwards compatible with all previous releases of that segment with exceptions, generate the Compat section of *SegInfo* by using the following format:

```
[Compat]
$EXCEPTIONS
version1[1]
version 2[1]
    •
    •
version n[1]
```

For segments which are backwards compatible with SPECIFIC previous releases of that segment, generate the Compat section of SegInfo by using the following format:

```
[Compat]
$LIST
version 1[1]
version 2[1]
    •
```

```
    •
version n[1]
```

NOTE: Append ":Pn" to specify specific patch compatibility, where Pn is the patch number that must be included with the current version of the segment to make it backwards compatible with the version that Pn has been appended to.

Refer to paragraph 5.5.5 of the DII COE I&RTS for more detailed information.

3.4.4.8 Conflicts

The Conflicts section of the SegInfo segment descriptor file is used to specify known inter-segment conflicts. The Conflicts section of the SegInfo segment descriptor file is optional for all segment types.

Step 1: Generate the Conflicts section of the *SegInfo* segment descriptor file by using the following format:

```
[Conflicts]
Segment name[1]: SegmentPrefix[2]:home dir[3] [:version[4]{:patch[5]}]
Segment name[1]: SegmentPrefix[2]:home dir[3] [:version[4]{:patch[5]}]
            •
            •
Segment name[1]: SegmentPrefix[2]:home dir[3] [:version[4]{:patch[5]}]
```

[1] ***Segment name*** = name of conflicting segment as determined by the SegName descriptor file.

[2] ***SegmentPrefix*** = conflicting segment's segment prefix.

[3] ***home dir*** = conflicting segment's home directory.

[4] ***version*** = specific version of conflicting segment.

[5] ***patch*** = specific patches of conflicting segment.

Refer to paragraph 5.5.6 of the DII COE I&RTS for more detailed information.

3.4.4.9 Data

The Data section of the SegInfo segment descriptor file is used to describe where data files are to be logically loaded and their scope (global, local, or segment).

Step 1: Generate the Data section of the SegInfo segment descriptor file using the following format:

```
[Data]
$SEGMENT[1]:segname[2]:segmentprefix[3]: home dir[4]
                OR
$LOCAL[5]:segname[2]:segmentprefix[3]: home dir[4]
                OR
$GLOBAL[6]:segname[2]:segmentprefix[3]: home dir[4]
```

[1] **$SEGMENT** = DII COE Keyword used when the data segment is local to the workstation and is managed and accessed by a single software segment.

[2] ***segname*** = string of up to 32 alphanumeric characters (spaces permitted) of the affected segment.

[3] ***segment prefix*** = string of up to 6 alphanumeric characters of the affected segment (no spaces, used in directory structure). Should be consistent with configuration management and segment registration process. Cannot use reserved segment prefixes identified in paragraph 5.3 of the DII COE I&RTS.

[4] ***home dir*** = "*/h/SegmentPrefix*" for the affected segment.

[5] **$LOCAL** = DII COE Keyword used when the data segment is limited to a local workstation but can be accessed by all users and applications local to that workstation.

[6] **$GLOBAL** = DII COE Keyword used when the data segment can be accessed by every workstation, every application and every operator having LAN access.

Refer to paragraph 5.5.7 of the DII COE I&RTS for more detailed information.

3.4.4.10 Database

This section of the SegInfo segment descriptor file is not currently implemented. It is reserved for future use to implement database specific requirements as they evolve.

3.4.4.11 Direct

The Direct section of the SegInfo segment descriptor is used to issue special instructions to the install software. The Direct section of the SegInfo segment descriptor file is optional for all segment types.

Step 1: Generate the Direct section of *SegInfo* by using the following format:

```
[Direct]
$ACCTADD[1]: executable[2]
$ACCTDEL[3]: executable[4]
$NOCOMPRESS[5]
$REBOOT[6]
$REMOTE[7] [:XTERM | :CHARBIF]
$ROOT[8]:PostInstall | PreInstall | DEINSTALL
```

[1] **$ACCTADD** = DII COE Keyword used to identify executables to be run upon adding an account.

[2] ***executable*** (ACCTADD) = executable file to be executed when an account is added.

[3] ***$ACCTDEL*** = DII COE Keyword used to identify executables to be run upon deleting an account.

[4] ***executable*** (ACCTDEL) = executable file to be executed when an account group is deleted.

[5] **$NOCOMPRESS** = DII COE Keyword used to suppress compression of the segment by the MakeInstall tool.

[6] **$REBOOT** = DII COE Keyword to have installation software perform an automatic reboot after the software installation is complete. The operator is provided an opportunity to override the reboot at install time.

[7] **$REMOTE** = DII COE Keyword to allow remote execution of the segment. This option can be directed to XTERM only or character interface only using the XTERM and CHARBIF keywords respectively.

[8] **$ROOT** = DII COE Keyword in the PostInstall, PreInstall and/or DEINSTALL must be run with root privileges.

Refer to paragraph 5.5.10 of the DII COE I&RTS for more detailed information.

3.4.4.12 FilesList

The FilesList file identifies all of the files and directories that a segment adds to the system. This FilesList Section of the SegInfo segment descriptor file is required only for COTS segment types.

Step 1: Generate the FilesList section of *SegInfo* by using the following format:

[FilesList]
$DIRS[1]
directory 1[2]
directory 2[2]
•
•
directory n[2]
$FILES[3]
file 1[4]
file 2[4]
•
•
file n[4]

[1] ***$DIRS*** = DII COE Keyword identifying segment directories

[2] ***directory*** = directories created by the segment. The $PATH variable can be set by including "$PATH:pathname" preceding the occurrence of the $DIRS. The $DIRS keyword must precede the list of directories.

[3] ***$FILES*** = DII COE Keyword to identify segment files.

[4] ***file*** = files added by the segment outside of the segments directory. All files under a segments individual directories are assumed to belong to the segment. The $PATH variable can be set by including $PATH:pathname preceding the occurrence of the $FILES. The $FILES keyword must precede the list of filenames.

Refer to paragraph 5.5.12 of the DII COE I&RTS for more detailed information.

3.4.4.13 Icons

The Icons section of the SegInfo segment descriptor file is used to identify the file in the segments data/icon directory that defines the icons that are made available on the desktop to launch segment functions. The Icons section of the SegInfo segment descriptor file is required for AcctGrp segment types and optional for COTS, Software, and Patch segment types.

Step 1: Generate the Icons section of the SegInfo segment descriptor file using the following format:

```
[Icons]
icon file[1]
```

[1] ***icon file*** = name of file in the Segments data/icons directory that associates segment executables with icons. The name can be up to 32 characters.

Step 2: Generate the Icon file for the segment in the ***SegmentPrefix\data\Icons*** directory using the following format:

```
window title[1]: icon path[2] : executable path[3] : comments[4]
```

[1] ***window title*** = title placed in the application window.

[2] ***icon path*** = file path to the icon image.

[3] ***executable path*** = full path of the executable to be launched by the menu program.

[4] ***comments*** = comment field to describe the icon .

Refer to paragraph 4.4.2 of the DII COE Programming Guide for more detailed information on adding icons to the COE. Refer to paragraph 5.5.14 of the DII COE I&RTS for additional information.

3.4.4.14 Menus

The Menus section of the SegInfo segment descriptor file is used to list the files under the data/menus directories defining menus to be used by and extended by the segment. The Menus section of the SegInfo segment descriptor file is required for AcctGrp segment types and optional for COTS, Software, and Patch segment types.

Step 1: For Account Group segments, generate the Menus section of the SegInfo segment descriptor file using the following format:

```
[Menus]
menu file 1[1]
menu file 2[1]
  •
  •
menu file n[1]
```

For COTS, Software, and Patch segments, generate the "Menus" section of the SegInfo segment descriptor file with the following format:

```
[Menus]
menu file 1[1]:[affected menu file[2]]
menu file 2[1]:[affected menu file[2]]
  •
  •
menu file n[1]:[affected menu file[2]]
```

[1] *menu file* = name of menu file listed in the segments data/menus directory.

[2] *affected menu file* = the name of the menu file under the account group's data/menus directory that is affected (replaced) by the segment's menu file. Affected menu file is optional.

The Menu section of the SegInfo segment descriptor file is used to add menu entries required by a segment. The entry in the SegInfo file refers to menu files generally contained in the "/h/*SegmentPrefix*/data/menus" directory. The menu files are formatted to pass menu information to the Common Desktop Environment (CDE) so that pull down menus, cascade menus, and menu items can be created and appended to. Refer to paragraph 4.4.1 of the DII COE Programming Guide for specific information on how to construct the menu files.

Refer to paragraph 5.5.16 of the DII COE I&RTS for more detailed information.

3.4.4.15 Network

The Network section of the SegInfo segment descriptor file is used to describe network-related parameters. The Network section of the SegInfo segment descriptor file is optional for segments with the COE CHILD or COE PARENT attributes.

Step 1: Generate the Network section of *SegInfo* by using the following format:

```
[Network]
$HOSTS[1]
LOCAL[2] | REMOTE[2] :IP address[3]:name[4]{:alias[5]}
$MOUNT[6]
hostname[7]:NFS mount point[8]:target dir[9]
$NETMASK[10]:mask[11]
$SERVERS[12]
server 1[13]
server 2[13]
  •
  •
server n[13]
```

[1] ***$HOSTS*** = DII COE Keyword used to establish host names.

[2] **LOCAL** is used if the entry is to be made only to the local hosts file. **REMOTE** is used if the entry is to made to the NIS+ or DNS server.

[3] ***IP address*** = used in the UNIX "/etc/hosts" file.

[4] ***name*** = used in the UNIX "/etc/hosts" file.

[5] ***alias*** = used in the UNIX "/etc/hosts" file.

[6] ***$MOUNT*** = DII COE Keyword used to set NFS mount points.

[7] ***hostname*** = name of the workstation on the network.

[8] ***NFS mount point*** = file partition to mount.

[9] ***target dir*** = where to mount the partition on the local machine.

[10] ***$NETMASK*** = DII COE Keyword used to establish network mask.

[11] ***mask*** = subnet mask.

[12] ***$SERVERS*** = DII COE Keyword used to establish servers providing COE services by symbolic name.

[13] ***server*** = symbolic name of the servers that the segment contains.

Refer to paragraph 5.5.19 of the DII COE I&RTS for more detailed information.

3.4.4.16 Permissions

The Permissions section of the SegInfo segment descriptor file is used to describe objects and permissions to grant or deny for the objects. The Permissions section of the SegInfo segment descriptor file is optional for AcctGrp, Software and Patch segment types.

Step 1: Generate the Permissions section of *SegInfo* by using the following format:

```
[Permissions]
object name 1[1]:permission abbreviation[2]:permission[3]
object name 2[1]:permission abbreviation[2]:permission[3]
                    •
                    •
object name n[1]:permission abbreviation[2]:permission[3]
```

[1] ***object name*** = item to be controlled.

[2] ***permission abbreviation*** = single character abbreviation for the permission (A = Add, D = Delete, E = Edit, P = Print, R = Read, V = View, X = Transmit). Additional abbreviations may be used as required.

[3] ***permission*** = permission type of access to grant or deny (Add, Delete, Read, etc...).

Refer to paragraph 5.5.20 of the DII COE I&RTS for more detailed information.

3.4.4.17 Processes

The Processes section of the SegInfo segment descriptor file is used to identify background processes for the segments. The Processes section of the SegInfo segment descriptor file is optional for AcctGrp, COTS, Software and Patch segment types.

Step 1: Generate the Processes section of SegInfo by using the following format:

```
[Processes]
$PATH[1]:pathname[2]
$BOOT[3]
process 1[4] {parameters[5]}
process 2[4] {parameters[5]}
            •
            •
process n[4] {parameters[5]}
$BACKGROUND[6]
process 1[4] {parameters[5]}
process 2[4] {parameters[5]}
            •
            •
process n[4] {parameters[5]}
$SESSION[7]
process 1[4] {parameters[5]}
```

process 2[4] *{parameters*[5]*}*
•
•
process n[4] *{parameters*[5]*}*
$SESSION_EXIT[8]
process 1[4] *{parameters*[5]*}*
process 2[4] *{parameters*[5]*}*
•
•
process n[4] *{parameters*[5]*}*

[1] ***$PATH*** = DII COE Keyword used to establish directory containing executables.

[2] ***pathname*** = the path of the directory containing the process file. Default pathname is segment's bin subdirectory. Default path is the segment bin directory.

[3] ***$BOOT*** = DII COE keyword used to define processes that launch at boot time.

[4] ***process*** = name of executable to launch.

[5] ***parameters*** = optional process dependent parameters.

[6] ***$BACKGROUND*** = DII COE keyword used to define background processes.

[7] ***$SESSION*** = DII COE keyword used to define login session processes.

[8] ***$SESSION_EXIT*** = DII COE keyword used to define processes that run prior to terminating a login session.

Refer to paragraph 5.5.24 of the DII COE I&RTS for more detailed information.

3.4.4.18 ReqrdScripts

The ReqrdScripts section of the SegInfo segment descriptor file is used to identify the script files that will be used to establish or extend the runtime environment. The ReqrdScripts section of the SegInfo segment descriptor file is required for Account Group segment types and optional for Software segment types.

For AcctGrp segment types, identify the script files that will be used to establish the runtime environment by generating the "ReqrdScripts" section with the following format:

[ReqrdScripts]
script name 1[1]:C[2] | L[3]
script name 2[1]:C[2] | L[3]
•
•
script name n[1]*:C*[2] *| L*[3]

[1] ***script name*** = name of the script in the affected account group's script subdirectory. Script name can be up to 32 characters.

[2] ***C*** = designates that the script file will be copied to the created user's login directory. These script files are to be located in the "/h/AcctGrps/AccountGroup/Scripts" directory where AccountGroup is the name of the Account Group. Script name can be up to 32 characters

[3] ***L*** = designates that the script file will be symbolically linked to the created user's login directory. These script files are to be located in the "/h/AcctGrps/AccountGroup/Scripts"

directory where AccountGroup is the name of the Account Group. Script name can be up to 32 characters

For Aggregate, COE Component, and Software segment types, identify the script files that will be used to extend the runtime environment by generating the "ReqrdScripts" section with the following format:

```
[ReqrdScripts]
script name 1[1]:env ext name[2]
script name 2[1]:env ext name[2]
        •
        •
script name n[1]:env ext name[2]
```

[1] ***script name*** = name of the script in the affected account group's script subdirectory. Script name can be up to 32 characters.

[2] ***env ext name*** = name of an environment extension file in the present segment's Scripts directory. Script name can be up to 32 characters.

Refer to paragraph 5.5.26 of the DII COE I&RTS for more detailed information.

3.4.4.19 Requires

The Requires section of the SegInfo segment descriptor file is used to identify segment dependencies. Although the Requires section of the SegInfo segment descriptor file is optional for all segment types, it is highly recommended that a statement indicating "no dependencies exist" if this is the case.

Step 1: Generate the Requires section of the SegInfo segment descriptor file by using the following format:

```
[Requires]
[$HOME_DIR[1]:pathname[2]]
segment name 1[3]: SegmentPrefix[4]:home dir[5]
segment name 2[3]: SegmentPrefix[4]:home dir[5]
                •
segment name n[3]: SegmentPrefix[4]:home dir[5]
```

[1] ***$HOME_DIR*** = DII COE Keyword to assign pathname.

[2] ***pathname*** = directory path of the segment home directory.

[3] ***segment name*** = name of the segment that must be loaded prior to the current segment.

[4] ***SegmentPrefix*** = prefix of the segment that must be loaded prior to the current segment.

[5] ***home dir*** = home directory of the segment that must be loaded prior to the current segment.

Refer to paragraph 5.5.27 of the DII COE I&RTS for more detailed information.

3.5 Creating Optional Segment Descriptor Files

This section describes the procedures for generating the optional segment descriptor files for all segment types. Optional segment descriptor files include DEINSTALL, FileAttribs, PostInstall, PreInstall, and PreMakeInstall. Table 3-4 shows the required and optional sections for each segment type.

Table 3-4: Segment Descriptor File Requirements per Segment Type

SegInfo Header	Software Segment	Account Group Segment	COTS Segment	Data Segment	Patch Segment
DEINSTALL	**N**	**R**	**N**	**N**	**N**
FileAttribs	**O**	**O**	**O**	**O**	**O**
Installed	**I**	**I**	**I**	**I**	**I**
PostInstall	**O**	**O**	**O**	**O**	**R**
PreInstall	**O**	**O**	**O**	**O**	**O**
PreMakeInst	**O**	**O**	**O**	**O**	**O**
ReleaseNotes	**R**	**R**	**R**	**R**	**R**
SegCheckSum	**I**	**I**	**I**	**I**	**I**
SegInfo	**R**	**R**	**R**	**R**	**R**
SegName	**R**	**R**	**R**	**R**	**R**
Validated	**I**	**I**	**I**	**I**	**I**
VERSIONS	**R**	**R**	**R**	**R**	**R**

R = Required, O = Optional, N = Not Required, I = Generated during Integration

3.5.1 DEINSTALL

The DEINSTALL segment descriptor file can be executed when an operator elects to remove a segment. It can be invoked by the operator to specifically remove a segment or invoked automatically when a segment is being updated. This DEINSTALL file can be a shell script created with an editor or an executable program. This file is not allowed to be run with root privileges. The DEINSTALL segment descriptor file is optional for all segment types.

Step 1: Generate the DEINSTALL segment descriptor file and include steps to invoke the scripts and/or executables required to remove all traces (files, directories) associated with the segment.

Step 2: Move the file to the "/h/SegmentPrefix/SegDescrip" directory (if necessary).

Refer to paragraph 5.5.9 of the DII COE I&RTS for more detailed information.

3.5.2 FileAttribs

The FileAttribs file is created by running the MakeAttribs tool. The MakeAttribs tool recursively traverses every subdirectory beneath the segment home directory and generates a line for each file that will be used to set the read/write/execute privileges for that file MakeAttribs does not include any file owned by root (A warning is printed). The Segment descriptor files under the SegDescrip directory are not included either. The format of each line is "permits:owner:group:filename". The installation tools will perform the following commands at install time for each line in FileAttribs:

chmod permits $INSTALL_DIR/filename

chown owner $INSTALL_DIR/filename

chgrp owner $INSTALL_DIR/filename

The FileAttribs segment descriptor file is optional for all segment types.

Step 1: Use the **MakeAttribs** tool to generate the FileAttribs segment descriptor file.

Step 2: Move the file to the "/h/*SegmentPrefix*/SegDescrip" directory (if necessary).

Refer to paragraph 5.5.11 and C.2.5 of the DII COE I&RTS for more detailed information.

3.5.3 PostInstall

The PostInstall segment descriptor file contains operations specific to installing the segment that must be performed after the segment software has been copied to the disk and installed by the COE software. The file can be a shell script file or an executable file. The PostInstall segment descriptor file is required for Patch type segments and optional for all other segment types.

Step 1: Generate the PostInstall segment descriptor file and include steps to invoke scripts and/or executables associated with the segment. **Do not** duplicate any operations performed by the COE installation software. Prompt the user as required.

Step 2: Move the file to the "/h/*SegmentPrefix*/SegDescrip" directory. (If necessary)

Refer to paragraph 5.5.21 of the DII COE I&RTS for more detailed information.

3.5.4 PreInstall

The PreInstall segment descriptor file contains operations specific to installing the segment that must be performed before the segment software has been copied to the disk. The file can be a shell script file or an executable file. The PreInstall segment descriptor file is optional for all segment types.

Step 1: Generate the PreInstall segment descriptor file and include steps to invoke scripts and/or executables associated with the segment. **Do not** duplicate any operations performed by the COE installation software. Prompt the user as required.

Step 2: Move the file to the "/h/*SegmentPrefix*/SegDescrip" directory (if necessary).

Refer to paragraph 5.5.22 of the DII COE I&RTS for more detailed information.

3.5.5 PreMakeInst

The PreMakeInst segment descriptor file is executed by the MakeInstall tool prior to creating the segment media. It contains "clean-up" commands for deleting temporary files and/or directories associated with the segment. The PreMakeInstall segment descriptor file is optional for all segment types.

Step 1: Generate the PreMakeInst segment descriptor file using the following format:

$PATH[1]*:pathname*[2]
rm tempfile 1[3]
rm tempfile 2[3]
•
•
rm tempfile n[3]
rmdir tempdir 1[4]
rmdir tempdir 2[4]
•

•
rmdir tempdir n[4]

[1] ***$PATH*** = DII COE keyword used to establish base directory where relative files and directories will be deleted from.

[2] ***pathname*** = directory path used to establish base directory where relative files and directories will be deleted from.

[3] ***tempfile*** = name of temporary files to be deleted.

[4] ***tempdir*** = name of temporary directory to be deleted.

Step 2: Move the file to the "/h/*SegmentPrefix*/SegDescrip" directory (if necessary).

Refer to paragraph 5.5.23 of the DII COE I&RTS for more detailed information.

3.6 Verifying the Segment

The tool **VerifySeg** must be run against all segments to validate that all segment descriptor files are complete and correct. VerifySeg must be rerun if any changes are made to any files (segment descriptor files, data files, executables, etc...) associated with the segment. VerifySeg will also be run by the COE Integrators to make sure that all submitted segments conform to the I&RTS.

Execute the **VerifySeg** tool to automatically generate the Validated descriptor file which includes the version of VerifySeg, the date and time of validation, who performed that validation, a count of errors and warnings, and a checksum using the following steps:

Step 1: Use the CalcSpace tool to update the value (in Kbytes) for $DISK in the Hardware section of SegInfo segment descriptor file using the following command sequence:

>> CalcSpace segmentdirectory

where *segmentdirectory* is the segment home directory under "/h".

Step 2: Type the following command sequence from the Unix prompt:

> VerifySeg segmentdirectory -p

where *segmentdirectory* is the segment home directory under under "/h".

Step 3: Review the results of the VerifySeg activity. If errors or warnings are indicated, changes will be required to the segment descriptor files and/or other files within the segment directory to resolve those errors and warnings.

Step 4: Rerun VerifySeg until all the errors = 0. Warnings are acceptable to continuing on with the segmentation effort but will impact the degree of COE runtime compliance that can be achieved by the segment.

NOTE: **DO NOT** edit a Segment Descriptor file without re-running VerifySeg.

Step 5: Review the file on the screen using "cat Validated" and verify that the information listed above is accurately contained in the file.

Refer to paragraph 5.5.33 of the DII COE I&RTS for more detailed information.

3.7 Testing Installation of Segments

Step 1: Run **TestInstall** using the following command (See Section C2.11 of the I&RTS for directions on running the tool):

>> TestInstall -p /h/segmentdirectory

3.8 Making Segment Install Media

This step is used create an installation tape of the segment if desired. This step is optional. (See paragraph C.2.6 of the I&RTS for additional information on using MakeInstall.) The tape can be created using the following steps:

Step 1: Use the following command to start the make install tape process:

>> MakeInstall -p /h segmentname

Step 2: Observe the following response:

Write to Disk
Write compressed to disk

list of devices available

Enter device to use (1, 2, etc) or type 'q' to quit.

Step 3: Type in the number of the device to be used:

>> device number

Step 4: Observe "Enter name of the output file or type 'q' to quit." on the display.

Step 5: Type in the name of the segment:

>> segmentname

Step 6: Observe the following on the display:

Processing Segment: /h/SegmentPrefix ...
Enter your name for the Tape Header:

Step 7: Type in the name of the segment:

>> segmentname

Step 8: Observe "Enter a serial number for the Tape Header:"

Step 9: Type in the following:

>> 1

Step 10: Observe the following on the display:

Enter any desired comment to put in the Tape Header (up to 255 characters) .

Step 11: Type in the following:

>> Test load of Segment segmentname

Step 12: Observe display showing the Tape Index, Attributes, Type, Hardware, Class, Home Directory and Segment Name.

Step 13: Observe display showing 1 segment to write to tape.

Step 14: Observe display showing space required in megabytes.

Step 15: Run **CanInstall** to verify that a segment is installable using the following command (See Appendix C of the I&RTS for directions on running the tool):

>> *CanInstall segmentname*

3.9 Performing System Test of Installed Segments

Step 1: Logon on to a Unix platform, with the DII COE installed, and logon as the system administrator.

Step 2: Hit a carriage return to get by the DISA disclaimer screen.

Step 3: Select the COEInstaller command from the Software pull-down menu on the display to Launch the COE Installer.

Step 4: Insert the first tape of the segmented application in the tape drive.

Step 5: Select the appropriate source drive from the COEInstaller window.

Step 6: Select the Read Table of Contents button from the COEInstaller window and wait for the contents of the diskette to show up in the COEInstaller window.

Step 7: Select the segmented application and verify that it shows as reverse video.

Step 8: Select the Release Notes button and verify that the release notes are displayed correctly.

Step 9: Select the Conflicts button and verify that conflicts identified during the segmentation process, if any, are displayed by COE Installer.

Step 10: Select the VERSION button and verify that version number and date identified during the segmentation process are displayed by COE Installer.

Step 11: Select the Install button and respond to any prompts by the segmented application's installation scripts.

Step 12: Verify that the COE Installer responds with a segment was successfully installed message and shows up in the "installed segments" area of the COEInstaller.

Step 13: If the installation failed, review the installation log via the pulldown menu and take appropriate action to resolve the problem. This may involve modification of the segmentation descriptor files, rerunning VerifySeg, TestInstall, TestRemove and MakeInstall to create a new version of the segment.

4. Segment Development - Windows NT Environment

The procedures in this section can be used to develop new segments from newly developed and/or migrated software applications that are intended to execute in a Windows NT runtime environment. These procedures assume a basic understanding of Windows NT commands and the Windows NT runtime environment. The process of segmenting an application is shown in Figure 4-1.

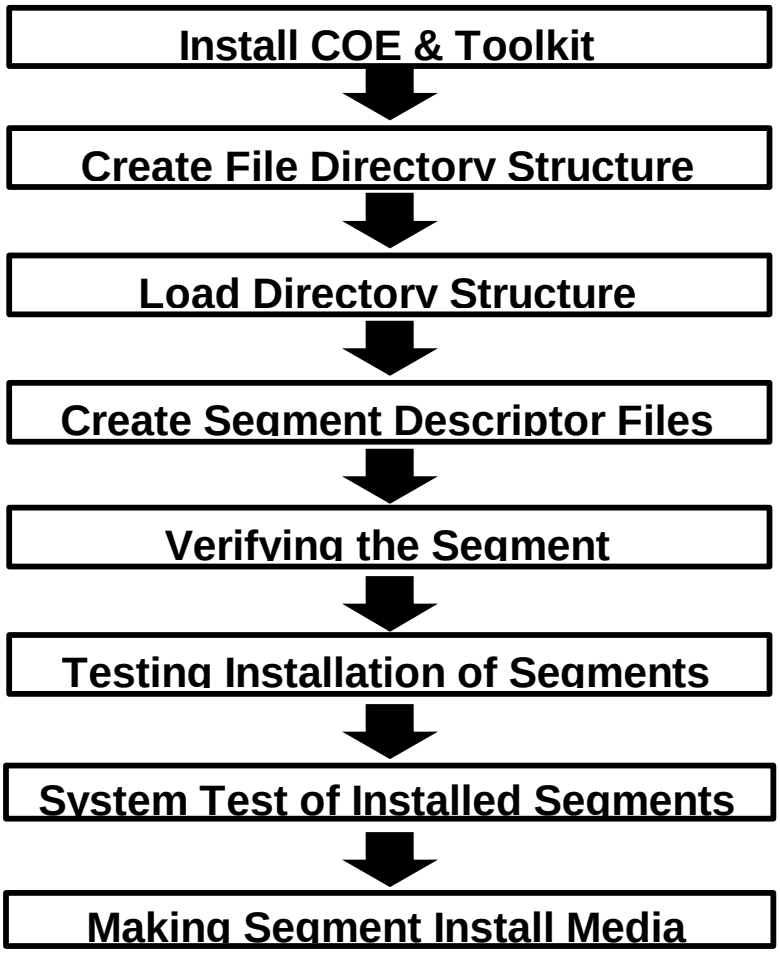

Figure 4-1: DII COE Segmentation Process

4.1 Installing the DII COE Windows NT Kernel and Developer's Toolkit

Table 4-1 identifies all of the available software diskettes for DII COE Release 2.0 for the Windows NT environment.

Table 4-1: DII COE Windows NT Release 2.0 Disks

Windows NT
DII COE Kernel Disks (5 for Release 2.0)
DII COE Windows NT Developer's ToolKit Disk
DII COE Segment Templates

The developer has two options prior to segmenting in establishing an environment for segmentation. The two options are:

- Load the DII COE Kernel **and** the Developer's Toolkit. Go to paragarph 4.1.1.

  OR

- Load **only** the Developer's Toolkit. The SegName segment descriptor file for the SysAdm account group must be created by the developer in order to successfully execute VerifySeg on any newly created segment.

  Create the following directory structure:

  c:\DII\Segments\SysAdm\SegDescrip

  Create the following SegName file within the above directory:

  $TYPE:ACCOUNT GROUP
  $NAME:System Administrator
  $PREFIX:SysAdm

  Skip to paragraph 4.1.3 of this document.

4.1.1 Installing the DII COE Windows NT Kernel

The developer must load the DII COE Kernel onto the development platform in order to have the ability to complete the development effort and to test the application software (once segmented) within the DII COE runtime environment.

NOTE: There should be at least 50 megabytes of avalable disk space identified on the platforms hard disk or a partition of a hard disk. This disk space must be formatted with the NTFS file system. The COEInstaller defaults to loading the DII COE Kernel onto the C: drive. This can be changed via the COEInstaller.

The DII COE for Windows NT is loaded in two parts:

- DII COE Kernel
- Additional DII COE segments if available

NOTE: There are no other segments available for DII COE Release 2.0 for the Windows NT.

The DII COE Kernel for Windows NT consists of the commercial software of the Windows NT Operating System, the System Administrator Account Group, and the COE Runtime Tool(s).

NOTE: The installer is strongly encouraged to install theWindows NT 3.51 operating system using the DII COE Kernel Installation Guide to ensure a known state for loading the remainder of the DII COE Kernel.

4.1.2 Verifying the Windows NT DII COE Successfully Loaded

Verify that the NT DII COE loaded by rebooting the system using the following steps:

Step 1: Login to Windows NT as administrator**.**

Step 2: Look for the System Administrator Icon in the Program Manager window**.**

Step 3: Look for the COE Installer Icon in the System Administrator window**.**

Step 4: Launch COE Installer and verify that it comes up successfully**.**

Step 5: Exit the COEInstaller**.**

4.1.3 Installing the DII COE Windows NT Developer's Toolkit

The tools required to support development and runtime capabilities make-up the DII COE Developer's Toolkit. Specifically, the DII COE Toolkit for COE Windows Nt release 2.0 contains the following:

- COE Development Tools (See I&RTS Appendix C)
  - Executable File: MakeInstall.exe
  - Executable File: TestInstall.exe
  - Executable File: TestRemove.exe
  - Executable File: VerifySeg.exe

- Other files in the MakeInstall folder.
  - setup.lib
  - MakeInst.ins

NOTE: All entries to be typed at the Microsoft Disk Operating System (MS DOS) command prompt or put into a Windows NT file are shown using *italics font* throughout this document. Items which are <u>underlined</u> require the developer to substitute segment specific information.

The C: disk drive is the default for loading the COE and it's tools. You may select another disk drive but for the purposes of this document we are assuming the default. The steps to install the DII COE Development Toolkit are as follows:

Step 1: Load the diskette into the 3 ½ (A:) disk drive of the development platform**.**

Step 2: Copy the tools from the A: drive to the C: drive using File Manager.

Step 3: Remove the disk from the 3 ½ (A:) disk drive of the development platform**.**

The directory structure created by extracting the Developer's Toolkit is shown in Figure 4-2. The executables for the tools that can be run are contained in the "/DII//tools/bin" subdirectory.

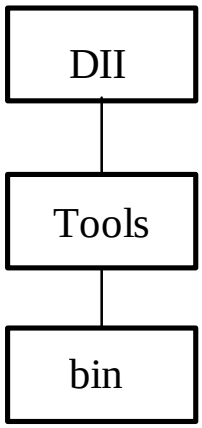

Figure 4-2: Developer's Toolkit Directory

4.2 Create Windows NT File Directory Structure

This document describes the establishment of a development environment by creating a directory structure and moving files to specific subdirectories. It provides the necessary environment to build the required segment descriptor files, icons, menus, fonts, application default files, data files, library files, and database management system (DBMS) files required for the segment. There are mandatory and optional subdirectories in the directory structure.

The directory structure is based on the runtime directory structure described in Chapter 5 of the DII I&RTS. The TOOLS subdirectory will have been created as a result of installing the Toolkit as described in paragraph 4.1, Install DII COE Developer's Toolkit.

Use the following steps to create a segment development directory structure as shown in Figure 4-3.

Step 1: Create a subdirectory for any and all segments being developed under "c:\Dev\Segments". The segment prefix identified in paragraph 2.9 is used as the subdirectory name and will be used later in many of the segment descriptor files.

Step 2: Create the "SegDescrip", "bin", "data", and "lib" subdirectories under each segment. The "lib" subdirectory is only required if public API libraries are being provided with the segment.

Step 3: Create the following subdirectories under c:\Dev\Segments*SegmentPrefix*\data if needed:

Icons
Menus
fonts
app-defaults

Step 4: If a database type segment is being developed, create the following subdirectories under c:\Dev\Segments*SegmentPrefix*

DBS_files
Install

Step 5: If a patch type segment is being developed, generate the following subdirectories under c:\Dev\Segments*SegmentPrefix*

Patchname.Pn

> **NOTE**: Append ":Pn" to specify specific patch compatibility, where Pn is the patch number that must be included with the current version of the segment to make it backwards compatible with the version that Pn has been appended to.

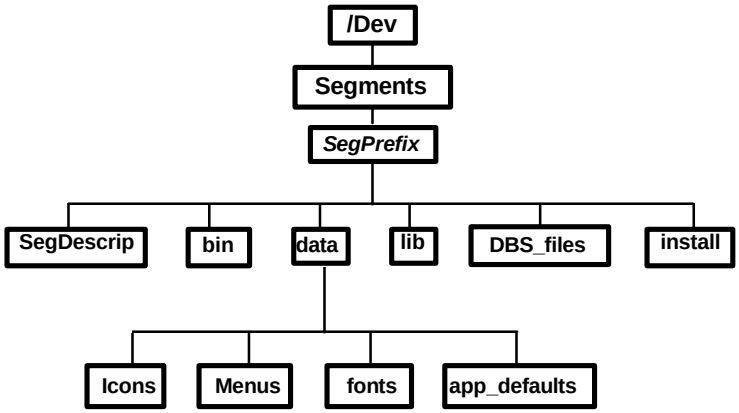

Figure 4-3: Segment Development Directory Structure

4.3 Load Windows NT Directories with Appropriate Files

Perform the following steps to load the COE development environment with the required files (Note: *SegmentPrefix* is the prefix assigned to the segment by the developer):

4.3.1 Executables Files

Copy executable files associated with the segment into "c:\Dev\Segments*SegmentPrefix*\bin".

4.3.2 Data Files

Copy data files associated with the segment into "c:\Dev\Segments*SegmentPrefix*\data". Data files may be in a directory structure under the "c:\Dev\Segments*SegmentPrefix*\data" directory.

4.3.3 lib Files

Copy public object code libraries (i.e. API's) associated with the segment into "/Dev/Segments/*SegmentPrefix*/lib". These files are not used at runtime but are provided to be available to developers through the future releases of the DII COE Developers Toolkit.

4.4 Create Required Segment Descriptor Files

This section describes the procedures for generating the **required** segment descriptor files for all segment types. The required segment descriptor files for all segmenttypes are SegName, ReleaseNotes, VERSION, and SegInfo.

Step 1: Get into the c:\Dev\Segments*SegmentPrefix*\SegDescrip directory for creating descriptor files.

Step 2: Create the segment descriptor files in following subparagraphs using an editor or by using the appropriate tools as specified.

4.4.1 SegName

The SegName segment descriptor file is required for all segment types. The SegName segment descriptor file contains information identifying the segment being created. Most of this information was determined back in paragraphs 2.6, 2.7, 2.8 and 2.9 of this document.

Step 1: Generate the SegName segment descriptor file using the following format:

NOTE: If the segment is a Patch segment then the "SegName" filename is followed by an extention of Pn (i.e., SegName.P1).

$TYPE[1]:segment type[2][:attribute[3]]
$NAME[4]:name[5]
$PREFIX[6]:SegmentPrefix[7]
$SEGMENT[8]:name[5]:SegmentPrefix[7]:home dir[9] ***OR***

$CHILD[10]:name[5]:SegmentPrefix[7]:home dir[9] ***OR***

$PARENT[11]:name[5]:SegmentPrefix[7]:home dir[9]

[1] ***$TYPE*** = DII COE Keyword sets segment type. The $TYPE keyword must be defined for all segment types.

[2] ***segment type*** = COTS, ACCOUNT GROUP, SOFTWARE, DATA, or PATCH. The COTS segment type is used if the segment is vendor provided. The ACCOUNT GROUP segment type is used if account specific icons, menus, and script files are being developed for a non-COE standard account group. The SOFTWARE segment type is used for mission application software or COE application software that is custom developed software. The SOFTWARE type is also used for Database segments for the time being. Future releases of the DII COE will support a DATABASE type. The DATA segment type is used if only local and global data files are being created/submitted as a segment. The PATCH segment type is used to create replacement files for segments that have already been delivered. Only those files that have been revised are submitted as a PATCH segment.

[3] ***attribute*** = AGGREGATE , CHILD, COE CHILD, COE PARENT. The AGGREGATE attribute is indicated if the segment being created is the parent segment of a group of mission application segments. The CHILD attribute is used if the segment is a child of a parent mission application segment. The COE CHILD attribute is used if the segment is a child of the parent COE segment. The COE PARENT attribute is used if the segment is to be the parent COE segment in a COE variant.

[4] **$NAME =** DII COE Keyword sets segment name. The $NAME keyword must be defined for all segment types.

[5] ***name*** **=** segment name consisting of a string of up to 32 alphanumeric characters (spaces permitted).

[6] **$PREFIX =** DII COE Keyword sets segment prefix. The $PREFIX keyword must be defined for all segment types.

[7] ***SegmentPrefix*** = segment prefix consisting of a string of up to 6 alphanumeric characters (no spaces, used in directory structure). Should be consistent with configuration management and segment registration process. Cannot use reserved segment prefixes identified in paragraph 5.3 of the DII COE I&RTS.

[8] **$SEGMENT** = DII COE Keyword in the file lists the affected segments or account groups that a patch segment applies to. This keyword is required if neither $CHILD or $PARENT are used. Multiple affected segments or child segments may be listed by listing each segment on a separate line..

[9] ***home dir*** = c:\DII\Segments*SegmentPrefix*

[10] **$CHILD** = DII COE Keyword is optional and lists the children if and only if the $AGGREGATE or $COE PARENT attributes have been listed. Multiple $CHILD lines can be used in this file.

[11] **$PARENT** = DII COE keyword is optional and lists the parent if and only if the $CHILD or $COE CHILD attributes have been listed. Only one line with $PARENT can be included in the file.

Step 2: Review the file on the screen and verify that the correct segment identification information is contained in the file.

Step 3: Move the SegName file to the "c:\DII\Segment\SegmentPrefix\SegDescrip" directory (if necessary).

Refer to paragraph 5.5.31 of the DII COE I&RTS for more detailed information.

4.4.2 VERSION

The VERSION segment descriptor file is required for all segment types. The VERSION segment descriptor file contains the time and date of when the segment was created. It must be created using an editor and is then updated by using the **TimeStamp** tool (currently not implemented in Windows NT releases of the DII COE).

Step 1: Generate the VERSION segment descriptor file using the following format:

version #[1]:date[2]

[1] ***version #*** = developer specified version number of the segment using the form a.b.c.d where a = major release; b = minor release; c = maintenance release; d = developer release. Refer to I&RTS paragraph 3.1 for additional information on version numbering.

[2] ***date*** = date the segment was created (or last modified) using the form mm/dd/yy.

Step 2: Move the VERSION file to the "c:\Dev\Segments\SegmentPrefix\SegDescrip" directory (if necessary).

Refer to paragraph 5.5.34 of the DII COE I&RTS for more detailed information.

4.4.3 ReleaseNotes

The ReleaseNotes segment descriptor file is required for all segment types. The ReleaseNotes descriptor file contains information identifying the new or enhanced functionality being provided by the segment.

Step 1: Generate the ReleaseNotes file and include information of interest to an operator. **Do not** include information on point of contacts, phone numbers, or help information. **Do** include known problems that have been fixed, new features introduced by this release, and version information. This file cannot include tabs or embedded control characters.

Step 2: Move the file to the "c:\Dev\Segments*SegmentPrefix*\SegDescrip" directory (if necessary).

Refer to paragraph 5.5.25 of the DII COE I&RTS for more detailed information.

4.4.4 SegInfo

The SegInfo segment descriptor file is required for all segment types. The SegInfo file contains several types of information used to integrate and install the segment. The SegInfo sections required depend upon the type of segment being created. Table 4-3 shows the required and optional sections within the SegInfo file for each segment type.

The Hardware and Security sections of SegInfo are required for all segment types. Some SegInfo sections described below are required, optional, or not required for some segment types. Each SegInfo section includes a heading in square brackets followed by keywords, commands, filenames, directory names, pathnames, etc. Not all commands are required within the section.

Table 4-2: SegInfo Section Dependencies per Segment Type

SegInfo Sections	Software Segment	Account Group Segment	COTS Segment	Data Segment	Patch Segment
AcctGroup	**N**	**R**	**N**	**N**	**N**
COEServices	**O**	**O**	**O**	**O**	**O**
Community					
Comm.deinstall					
Compat	**O**	**O**	**O**	**O**	**O**
Conflicts	**O**	**O**	**O**	**O**	**O**
Data	**N**	**N**	**N**	**R**	**N**
Database	**Reserved**	**Reserved**	**Reserved**	**Reserved**	**Reserved**
Direct	**O**	**O**	**O**	**O**	**O**
FilesList	**O**	**O**	**R**	**O**	**O**
Hardware	**R**	**R**	**R**	**R**	**R**
Icons	**O**	**R**	**O**	**N**	**O**
Menus	**O**	**R**	**O**	**N**	**O**
Network					
Permissions	**O**	**O**	**N**	**N**	**O**
Processes	**O**	**O**	**O**	**N**	**O**
Requires	**O**	**O**	**O**	**O**	**O**
RqrdScrpts					
Security	**R**	**R**	**R**	**R**	**R**

R = Required, O = Optional, N = Not Applicable

NOTE: This table assumes level 5 runtime(I&RTS) compliance. Items grayed out are not implemented for Windows NT COE at this time.

4.4.4.1 Hardware

The Hardware section of SegInfo is required for all segment types. It specifies the computing resources required by the segment. Specifically, it identifies the platform using a predefined keyword, the RAM in kilobytes, the amount of disk space and growth disk space in kilobytes, the disk partitions and growth disk partitions required in kilobytes, the operating system using a predefined keyword, and the amount of temporary space required during installation.

Step 1: Generate the Hardware section of SegInfo using the following format:

[Hardware]
$CPU[1]:platform[2]
$MEMORY[3]:size[4]
$DISK[5]:size[6][:reserve][7] ***or*** *$PARTITION[8]:diskname[9]:size[10][:reserve[7]]*
•
•
•
$PARTITION[8]:diskname[9]:size[10][:reserve[7]]
$OPSYS[11]:operating system[12]
$TEMPSPACE[13]:size[14]

[1] **$CPU** = DII COE keyword establishes platform type. The $CPU keyword must be defined for all segment types.

[2] ***platform*** = target runtime platform dependency identified as one of the following:

ALL:	Platform Independent
PC:	All PC platforms that support NT
PC486:	Defined for INTEL PC486 workstations
PENTIUM:	Defined for INTEL PENTIUM workstations

[3] **$MEMORY** = DII COE keyword defines segment memory requirements. The $MEMORY keyword is required for all segments except for DATA segments.

[4] ***size*** **(MEMORY)** = amount of RAM required by the segment in Kilobytes

[5] **$DISK** = DII COE keyword defines segment disk requirements.

[6] ***size*** **(DISK)** = size of the segment (and all subdirectories) at install time expressed in Kilobytes. Once this value is established, the COE tool CalcSpace can be used to automatically calculate the size of the segment and update the $DISK keyword accordingly.

[7] ***reserve*** = amount of extra disk space in Kilobytes reserved to accommodate future growth of the segment.

[8] **$PARTITION** = DII COE keyword defines segment partitions.

[9] ***diskname*** = explicit partition name (e.g., /home2) or an environment variable name of the form DISK1, DISK2, ...DISK99. The installation software will set the environment variables DISK1, DISK2, etc... to the absolute pathname where space has been allocated. $PARTION keywords are assumed to be in sequential order.

[10] ***size*** **(PARTITION)** = size of the segment in Kilobytes on a particular disk partition. Installation software does not allow a segment to be split across multiple disk partitions. However, the PostInstall script does allow for splitting the segment across multiple disk partitions.

[11] **$OPSYS** = DII COE keyword defines operating system requirement of the segment.

[12] ***operating system*** = DII COE supported operating systems (as listed in I&RTS paragraph 5.3) for MACHINE_CPU. The only operating system for the PC is NT.

[13] **$TEMPSPACE** = DII COE keyword defines temporary space requirements of the segment.

[14] ***size*** **(TEMPSPACE)** = amount of temporary diskspace in kilobytes that are used during the installation process. If space is available, the installation software sets the variable COE_TMPSPACE to the absolute pathname where space is allocated. If not enough space can be found, an error message is displayed to the operator.

Refer to paragraph 5.5.13 of the DII COE I&RTS for more detailed information.

4.4.4.2 Security

The Security section of the SegInfo segment descriptor file is required for all segment types. The Security section of the SegInfo segment descriptor file indicates the highest classification level for the segment.

Step 1: Generate the Security section of the SegInfo descriptor file using the following format:

```
[Security]
classification level[1]
```

[1] ***classification level*** = UNCLASS, CONFIDENTIAL, SECRET, or TOP SECRET indicating the segment classification level of the segment.

Refer to paragraph 5.5.28 of the DII COE I&RTS for more detailed information.

4.4.4.3 AcctGroup

The AcctGroup section of the SegInfo descriptor file is required for all Account Group type segments.

Step 1: Generate the AcctGroup section of the SegInfo segment descriptor file using the following format:

```
[AcctGroup]
$group name[1]:group ID[2]:: profile flag[3]:home dir[4]:default profile name[5]
$CLASSIF[6]:classification[7]
```

Note: Two "colons" after the group ID filed are required.

[1] ***group name*** = Alphanumeric string up to 15 characters used to identify this account group. The account group name must be unique (i.e., no other account group may have the same name).

[2] ***group id*** = Group ID to be inserted into the password file for accounts created from this group.

[3] ***profile flag*** = 0 if no profiles are allowed, otherwise 1.

[4] ***home dir*** = Home directory for the account group (e.g., *c:\DII\Segments\SysAdm*). The path name may contain up to 256 characters. SysAdm is currently the only one available with the DII COE Kernel.

[5] ***default profile name*** = Alphanumeric string up to 15 characters identifying the account group's default profile. This name is ignored unless the profile flag is non-zero.

[6] **$CLASSIF** = DII COE keyword to specify classification of an Account Group Sgement.

[7] ***Classification level*** = classification of the profile and can be UNCLASS, CONFIDENTIAL, SECRET, or TOP SECRET. The classification level for a profile will default to TOP SECRET unless the segment has defined it otherwise.

Refer to paragraph 5.5.1 of the DII COE I&RTS for more detailed information.

4.4.4.4 COEServices

The COEServices section of the SegInfo segment descriptor file is used to specify changes in services provided by the operating system. The COEServices section of the SegInfo segment descriptor file is optional for all segment types.

Step 1: Generate the COEServices section of the SegInfo segment descriptor file using the following format:

```
[COEServices]
$SERVICES[:servicescomment[1]]
name:port[2]:protocol[3]{:alias[4]}
name:port[2]:protocol[3]{:alias[4]}
            •
            •
socketname[5]:port[2]:protocol[3]{:alias[4]}
```

[1] ***servicescomment*** = comment included in the "/etc/services" system file for the port.

[2] ***port*** = port number requested. Numbers 2000-2999 are reserved for COE segments.

[3] ***protocol*** = tcp or udp.

[4] ***alias*** = the symbolic name used to refer to the assigned port.

[5] ***socketname*** = name of the socket add to the "/etc/services" system file.

Refer to paragraph 5.5.2 of the DII COE I&RTS for more detailed information.

4.4.4.5 Community

Not implemented in NT at this time.

4.4.4.6 Comm.deinstall

Not implemented in NT at this time.

4.4.4.7 Compat

The Compat section of the *SegInfo* segment descriptor file is used to indicate the degree to which backward compatibility is preserved with the newly released segment. The Compat section of the *SegInfo* segment descriptor file is optional for all segment types.

Step 1: For segments which are backwards compatible with all previous releases of that segment, generate the Compat section of *SegInfo* by using the following format:

```
[Compat]
+ALL
```

For segments which are not backwards compatible with any previous releases of that segment, generate the Compat section of *SegInfo* by using the following format:

```
[Compat]
-NONE
```

For segments which are backwards compatible with specific previous releases of that segment starting from an earliest release, generate the Compat section of *SegInfo* by using the following format:

```
[Compat]
$EARLIEST
version[1]
```

[1] ***version*** = specific version number of the DII COE version format "a.b.c.d".

For segments which are backwards compatible with all previous releases of that segment with exceptions, generate the Compat section of *SegInfo* by using the following format:

```
[Compat]
$EXCEPTIONS
version1[1]
version2[1]
      •
      •
versionn[1]
```

[1] ***version*** = specific version number of the DII COE version format "a.b.c.d".

For segments which are backwards compatible with SPECIFIC previous releases of that segment, generate the Compat section of SegInfo by using the following format:

```
 [Compat]
$LIST
version1[1]
version 2[1]
                •
                •
version n[1]
```

[1] ***version*** = specific version number of the DII COE version format "a.b.c.d".

NOTE: Append ":Pn" to specify specific patch compatibility, where Pn is the patch number that must be included with the current version of the segment to make it backwards compatible with the version that Pn has been appended to.

Refer to paragraph 5.5.5 of the DII COE I&RTS for more detailed information.

4.4.4.8 Conflicts

The Conflicts section of the SegInfo segment descriptor file is used to specify known inter-segment conflicts. The Conflicts section of the SegInfo segment descriptor file is optional for all segment types.

Step 1: Generate the Conflicts section of the *SegInfo* segment descriptor file by using the following format:

```
[Conflicts]
Segment name[1]: SegmentPrefix[2]:home dir[3] [:version[4]{:patch[5]}]
Segment name[1]: SegmentPrefix[2]:home dir[3] [:version[4]{:patch[5]}]
                •
                •
Segment name[1]: SegmentPrefix[2]:home dir[3] [:version[4]{:patch[5]}]
```

[1] ***Segment name*** = name of conflicting segment as determined by the SegName descriptor file.

[2] ***SegmentPrefix*** = conflicting segment's segment prefix.

[3] ***home dir*** = conflicting segment's home directory.

[4] ***version*** = specific version of conflicting segment.

[5] ***patch*** = specific patches of conflicting segment.

Refer to paragraph 5.5.6 of the DII COE I&RTS for more detailed information.

4.4.4.9 Data

The Data section of the SegInfo segment descriptor file is required for data segments and is used to describe where data files are to be logically loaded and their scope (global, local, or segment).

Step 1: Generate the Data section of the SegInfo segment descriptor file using the following format:

[Data]
$SEGMENT[1]:segname[2]:segmentprefix[3]: home dir[4]
OR
$LOCAL[5]:segname[2]:segmentprefix[3]: home dir[4]
OR
$GLOBAL[6]:*segname[2]: segmentprefix[3]: home dir[4]*

[1] **$SEGMENT** = DII COE keyword used when the data segment is local to the workstation and is managed and accessed by a single software segment.

[2] ***segname*** = string of up to 32 alphanumeric characters (spaces permitted) of the affected segment.

[3] ***segmentprefix*** = string of up to 6 alphanumeric characters of the affected segment (no spaces, used in directory structure). Should be consistent with configuration management and segment registration process. Cannot use reserved segment prefixes identified in paragraph 5.3 of the DII COE I&RTS.

[4] ***home dir*** = "*c:\DII\Segments\segmentprefix*" for the affected segment.

[5] **$LOCAL** = DII COE keyword used when the data segment is limited to a local workstation but can be accessed by all users and applications local to that workstation.

[6] **$GLOBAL** = DII COE keyword used when the data segment can be accessed by every workstation, every application and every operator having LAN access.

Refer to paragraph 5.5.7 of the DII COE I&RTS for more detailed information.

4.4.4.10 Database

This section of the SegInfo segment descriptor file is not currently implemented. It is reserved for future use to implement database specific requirements as they evolve.

4.4.4.11 Direct

The Direct section of the SegInfo segment descriptor is used to issue special instructions to the install software. The Direct section of the SegInfo segment descriptor file is optional for all segment types.

Step 1: Generate the Direct section of *SegInfo* by using the following format:

[Direct]
$ACCTADD[1]*:* *executable*[2]
$ACCTDEL[3]*:* *executable*[4]
$NOCOMPRESS[5]
$REBOOT[6]
$REMOTE[7] *[:XTERM | :CHARBIF]*
$ROOT[8]*:PostInstall | PreInstall | DEINSTALL*

[1] **$ACCTADD** = DII COE Keyword used to identify executables to be run upon adding an account.

[2] ***executable*** (ACCTADD) = executable file to be executed when an account is added.

[3] **$ACCTDEL** = DII COE Keyword used to identify executables to be run upon deleting an account.

[4] ***executable*** (ACCTDEL) = executable file to be executed when an account group is deleted.

[5] **$NOCOMPRESS** = DII COE Keyword used to suppress compression of the segment by the MakeInstall tool.

[6] **$REBOOT** = DII COE Keyword to have installation software perform an automatic reboot after the software installation is complete. The operator is provided an opportunity to override the reboot at install time.

[7] **$REMOTE** = DII COE Keyword to allow remote execution of the segment. This option can be directed to XTERM only or character interface only using the XTERM and CHARBIF keywords respectively.

[8] **$ROOT** = DII COE Keyword in the PostInstall, PreInstall and/or DEINSTALL must be run with administrator privileges.

Refer to paragraph 5.5.10 of the DII COE I&RTS for more detailed information.

4.4.4.12 FilesList

The FilesList section identifies all the files and directories that a segment adds to the system. This FilesList section of the SegInfo segment descriptor file is required only for COTS segment types.

Step 1: Generate the FilesList section of *SegInfo* by using the following format:

[FilesList]
$DIRS[1]
directory 1[2]
directory 2[2]
•
•
•
directory n[2]

$FILES[3]
file 1[4]
file 2[4]
•
•
•
file n[4]

[1] **$DIRECTORY** = DII COE keyword identifies segment directories

[2] ***directory*** = directories to be created by the segment. The $PATH variable can be set by including "$PATH:pathname" preceding the occurrence of the $DIRS. The $DIRS keyword must precede the list of directories.

[3] **$FILES** = DII COE keyword identifies segment files.

[4] ***file*** = files that are to be added by the segment outside of the segments directory. All files under a segments individual directories are assumed to belong to the segment. The $PATH variable can be set by including $PATH:pathname preceding the occurrence of the $FILES. The $FILES keyword must precede the list of filenames.

Refer to paragraph 5.5.12 of the DII COE I&RTS for more detailed information.

4.4.4.13 Icons

The Icons section of the SegInfo segment descriptor file is used to identify the file in the segments data/icon directory that defines the icons that are made available on the desktop to launch segment functions. The Icons section of the SegInfo segment descriptor file is required for AcctGrp segment types and optional for COTS, Software, and Patch segment types.

Step 1: Generate the Icons section of the SegInfo segment descriptor file using the following format:

[Icons]
icon file[1]

[1] ***icon file*** = name of file in the Segments data/icons directory that associates segment executables with icons. The name can be up to 32 characters.

Step 2: Generate the Icon file for the segment in the ***SegmentPrefix\data\Icons*** directory using the following format:

window title[1] : *icon path*[2] : *executable path*[3] : *comments*[4]

[1] ***window title*** = title placed in the application window.

[2] ***icon path*** = file path to the icon image. It should be the same as the executable path for the Windows NT COE.

[3] ***executable path*** = full path of the executable to be launched by the menu program.

[4] ***comments*** = comment field to describe the icon .

Refer to paragraph 4.4.2 of the DII COE Programming Guide for more detailed information on adding icons to the COE. Refer to paragraph 5.5.14 of the DII COE I&RTS for additional information.

4.4.4.14 Menus

Not implemented in NT at this time.

4.4.4.15 Permissions

The Permissions section of the SegInfo segment descriptor file is used to describe objects and permissions to grant or deny for the objects. The Permissions section of the SegInfo segment descriptor file is optional for AcctGrp, Software and Patch segment types.

Step 1: Generate the Permissions section of *SegInfo* by using the following format:

[Permissions]
object name 1[1]*:permission abbreviation*[2]*:permission*[3]
object name 2[1]*:permission abbreviation*[2]*:permission*[3]
•
•
object name n[1]*:permission abbreviation*[2]*:permission*[3]

[1] ***object name*** = item to be controlled.

[2] ***permission abbreviation*** = single character abbreviation for the permission (A = Add, D = Delete, E = Edit, P = Print, R = Read, V = View, X = Transmit). Additional abbreviations may be used as required.

[3] ***permission*** = permission type of access to grant or deny (Add, Delete, Read, etc...).

Refer to paragraph 5.5.20 of the DII COE I&RTS for more detailed information.

4.4.4.16 Processes

Not implemented in NT at this time.

4.4.4.17 Network

Not implemented in NT at this time.

4.4.4.18 Permissions

The Permissions section of the SegInfo segment descriptor file is used to describe objects and permissions to grant or deny for the objects. The Permissions section of the SegInfo segment descriptor file is optional for AcctGrp, Software and Patch segment types.

Step 1: Generate the Permissions section of *SegInfo* by using the following format:

[Permissions]
object name 1[1]*:permission abbreviation*[2]*:permission*[3]
object name 2[1]*:permission abbreviation*[2]*:permission*[3]
•
•
object name n[1]*:permission abbreviation*[2]*:permission*[3]

[1] ***object name*** = item to be controlled.

[2] ***permission abbreviation*** = single character abbreviation for the permission (A = Add, D = Delete, E = Edit, P = Print, R = Read, V = View, X = Transmit). Additional abbreviations may be used as required.

[3] ***permission*** = permission type of access to grant or deny (Add, Delete, Read, etc...).

4.4.4.19 ReqrdScripts

Not implemented in NT at this time.

4.4.4.20 Requires

The Requires section of the SegInfo segment descriptor file is used to identify segment dependencies. Although the Requires section of the SegInfo segment descriptor file is optional for all segment types, it is highly recommended that a statement indicating "no dependencies exist" if this is the case.

Step 1: Generate the Requires section of the SegInfo segment descriptor file by using the following format:

[Requires]
[*$HOME_DIR*[1]:*pathname*[2]]
segment name 1[3]: *SegmentPrefix*[4]:*home dir*[5]
segment name 2[3]: *SegmentPrefix*[4]:*home dir*[5]
•
•
segment name n[3]: *SegmentPrefix*[4]:*home dir*[5]

[1] ***$HOME_DIR*** = optional keyword to assign pathname.

[2] ***pathname*** = directory path of the segment home directory.

[3] ***segment name*** = name of the segment that must be loaded prior to the current segment.

[4] ***SegmentPrefix*** = prefix of the segment that must be loaded prior to the current segment.

[5] ***home dir*** = home directory of the segment that must be loaded prior to the current segment.

Refer to paragraph 5.5.27 of the DII COE I&RTS for more detailed information.

4.5 Creating Optional Segment Descriptor Files

This section describes the procedures for generating the optional segment descriptor files for all segment types. Optional segment descriptor files include DEINSTALL, FileAttribs, PostInstall, PreInstall, and PreMakeInstall. Table 4-2 shows the required and optional sections for each segment type.

Table 4-3: Segment Descriptor File Requirements per SegmentType

Segment Descriptor Files	Software Segment	Account Group Segment	COTS Segment	Data Segment	Patch Segment
DEINSTALL.bat DEINSTALL.exe	**O**	**O**	**O**	**O**	**O**
FileAttribs	**O**	**O**	**O**	**O**	**O**
Installed	**I**	**I**	**I**	**I**	**I**
PostInstall.bat PostInstall.exe	**O**	**O**	**O**	**O**	**R**
PreInstall.bat PreInstall.exe	**O**	**O**	**O**	**O**	**O**
PreMakeInst.bat PreMakeInst.exe	**O**	**O**	**O**	**O**	**O**
ReleaseNotes	**R**	**R**	**R**	**R**	**R**
SegCheckSum	**I**	**I**	**I**	**I**	**I**
SegInfo	**R**	**R**	**R**	**R**	**R**
SegName	**R**	**R**	**R**	**R**	**R**
Validated	**I**	**I**	**I**	**I**	**I**
VERSIONS	**R**	**R**	**R**	**R**	**R**

R = Required, O = Optional, I = Generated during Integration

4.5.1 DEINSTALL

The DEINSTALL segment descriptor file can be executed when an operator elects to remove a segment. It can be invoked by the operator to specifically remove a segment or invoked automatically when a segment is being updated. The DEINSTALL file can be a shell script created with an editor or an executable program. The DEINSTALL segment descriptor file is optional for all segment types however, if this file does not exist, then the segment cannot be removed with the COEInstaller. In order for the COEInstaller to be able to deinstall a segment, a deinstall file must exist and contain at least one line so as not to have an empty file.

Step 1: Generate the DEINSTALL segment descriptor file and include steps to invoke the scripts and/or executables required to remove all traces (files, directories) associated with the segment.

Step 2: Move the file to the "c:\DII\Segments\SegmentPrefix\SegDescrip" directory (if necessary).

Refer to paragraph 5.5.9 of the DII COE I&RTS for more detailed information.

4.5.2 PostInstall

The PostInstall segment descriptor file contains operations specific to installing the segment that must be performed after the segment software has been copied to the disk and installed by the COE software. The file can be a shell script file or an executable file. The PostInstall segment descriptor file is required for Patch type segments and optional for all other segment types.

Step 1: Generate the PostInstall segment descriptor file and include steps to invoke scripts and/or executables associated with the segment. **Do not** duplicate any operations performed by the COE installation software. Prompt the user as required.

Step 2: Move the file to the "c:\DII\Segments\SegmentPrefix\SegDescrip" directory (if necessary).

Refer to paragraph 5.5.21 of the DII COE I&RTS for more detailed information.

4.5.3 PreInstall

The PreInstall segment descriptor file contains operations specific to installing the segment that must be performed before the segment software has been copied to the disk. The file can be a shell script file or an executable file. The PreInstall segment descriptor file is optional for all segment types.

Step 1: Generate the PreInstall segment descriptor file and include steps to invoke scripts and/or executables associated with the segment. **Do not** duplicate any operations performed by the COE installation software. Prompt the user as required.

Step 2: Move the file to the "c:\DII\Segments*SegmentPrefix*\SegDescrip" directory (if necessary).

Refer to paragraph 5.5.22 of the DII COE I&RTS for more detailed information.

4.5.4 PreMakeInst

The PreMakeInst segment descriptor file is executed by the MakeInstall tool prior to creating the segment media. It contains "clean-up" commands for deleting temporary files and/or directories associated with the segment. The PreMakeInstall segment descriptor file is optional for all segment types.

Step 1: Generate the PreMakeInst segment descriptor file using the following format:

```
$PATH[1]:pathname[2]
rm tempfile 1[3]
rm tempfile 2[3]
        •
        •
rm tempfile n[3]
rmdir tempdir 1[4]
rmdir tempdir 2[4]
        •
        •
rmdir tempdir n[4]
```

[1] **$PATH** = DII COE keyword establishes base directory where relative files and directories will be deleted from.

[2] ***pathname*** = directory path establishes base directory where relative files and directories will be deleted from.

[3] ***tempfile*** = name of temporary files to be deleted.

[4] ***tempdir*** = name of temporary directory to be deleted.

Step 2: Move the file to the "c:\DII\SegmentS*SegmentPrefix*\SegDescrip" directory (if necessary).

Refer to paragraph 5.5.23 of the DII COE I&RTS for more detailed information.

4.6 Verifying the Segment

The tool **VerifySeg** must be run against all segments to validate that all associated segment descriptor files are complete and correct. VerifySeg must be rerun if any changes are made to any files (segment descriptor files, data files, executables, etc.) associated with the segment. VerifySeg will also be run by the COE Integrators to ensure all submitted segments conform to the I&RTS.

Execute the **VerifySeg** tool to automatically generate the Validated descriptor file which includes the version of VerifySeg, the date and time of validation, who performed that validation, a count of errors and warnings, and a checksum using the following steps:

Step 1: Type the following command sequence from the MS DOS prompt:

> VerifySeg segmentdirectory -p

where *segmentdirectory* is the segment home directory under "c:\DII\Segments".

Step 2: Review the results of the VerifySeg activity. If errors or warnings are indicated, changes will be required to the segment descriptor files and/or other files within the segment directory to resolve those errors and warnings.

Step 3: Rerun VerifySeg until all the errors = 0. Warnings are acceptable to continuing on with the segmentation effort but will impact the degree of COE runtime compliance that can be achieved by the segment.

NOTE: **DO NOT** edit a Segment Descriptor file without re-running VerifySeg.

Refer to paragraph 5.5.33 of the DII COE I&RTS for more detailed information.

4.7 Testing that the Segment is Installable

The tape can be created using the following steps:

Step 1: Run **TestInstall** using the following command (See Section C2.11 of the I&RTS for directions on running the tool) from the DOS prompt:

>> TestInstall -p /h/segmentdirectory segment name

Step 2: Select continue when prompted to continue with the COE TestInstall

4.8 Making Segment Install Media

This step is optional and used to create an installation disk(s) of the segment if desired. See paragraph C.2.6 of the I&RTS for additional information on using MakeInstall. The tape can be created using the following steps:

Step 1: Type the following command from the MS DOS prompt to start the make install tape process:

> *MakeInstall -p c:\DII\Segments segmentname*

Step 2: When the "Make Install" window appears enter the Creator, Serial Number and any Comments in the appropriate area then click on the "Next" button.

Step 3: "Processing Segment: *SegmentPrefix*" message will appear. Answer YES to question "Do you want to erase all data?" to continue.

NOTE: MakeInstall will erase all data previously saved on the disk(s).

Step 4: Follow instructions for inserting disk(s) into Drive A:.

Step 5: Upon a successful MakeInstall the following is displayed:

Segment Index
Segment Name
Segment Path
Segment Version
Segment Attribute
Segment Type
Segment Hardware
Segment Security
Start Disk
End Disk

Number of Segments to write to Disk

Please click on "Finish to MakeInstall" button.

4.9 Performing System Test of Installed Segments

Verify that the segmented application can be loaded by the COEInstaller and function correctly in the COE runtime environment. The actual functional and performance testing specific to the segmented application and thus it is left up to the developer to identify the appropriate test procedures. As a minimum, the developer should verify that the segment is installable and that it can be launched from the DII COE desktop. In the case of the COE for the Windows NT, launching the segmented application will be done via the native Windows NT desktop.

Install the segmented application via the COEInstaller by performing the following steps on a Windows NT platform with the DII COE installed:

Step 1: Logon on to a Windows NT platform, with the DII COE installed, as the system administrator.

Step 2: Select the System Administrator Window via the System Administrator Icon in the Program Manager window**.**

Step 3: Launch the COEInstaller from the System Administrator window**.**

Step 4: Hit a carriage return to get by the DISA disclaimer screen.

Step 5: Insert the first diskette of the segmented application in the disk drive.

Step 6: Select the appropriate source drive (usually A:) from the COEInstaller window.

Step 7: Select the Read Table of Contents button from the COEInstaller window and wait for the contents of the diskette to show up in the COEInstaller window.

Step 8: Select the segmented application and verify that it shows as reverse video.

Step 9: Select the Release Notes button and verify that the release notes are displayed correctly.

Step 10: Select the Conflicts button and verify that conflicts identified during the segmentation process, if any, are displayed by COE Installer.

Step 11: Select the VERSION button and verify that version number and date identified during the segmentation process are displayed by COE Installer.

Step 12: Select the Install button and respond to any prompts by the segmented application's installation scripts.

Step 13: Verify that the COE Installer responds with a segment was successfully installed message and shows up in the "installed segments" area of the COEInstaller.

Step 14: If the installation failed, review the installation log via the pull-down menu and take appropriate action to resolve the problem. This may involve modification of the segmentation descriptor files, rerunning VerifySeg, TestInstall, TestRemove and MakeInstall to create a new version of the segment.

Step 15: Once the segmented application successfully installed, exit the COEInstaller.

Step 16: Launch the segment from the native Windows NT desktop.

Step 17: Verify that the segment launched successfully and start perform functional and performance testing appropriate for the specific segment

Step 18: Launch the COEInstaller from the System Administrator Window.

Step 19: Select the segmented application from the installed segments are of the COE Installer window.

Step 20: Select the DEINSTALL button from the COE Installer and verify that the DEINSTALL.exe or DEINSTALL.bat executes.

Step 21: Select the DEINSTALL button from the COE Installer and verify that the DEINSTALL.exe or DEINSTALL.bat executes.

Step 22: Once the segmented application successfully deinstalled, exit the COEInstaller.

Step 23: Using the file manager, verify that the icons and/or menus associated with the segmented application are no longer installed.

5. Submitting Segments for Integration and Testing

Segments are formally delivered to either the DISA (DCTF) or the Operational Support Facility (OSF). GCSS mission application segments are delivered to the DCTF. GCCS mission application segments and DII COE component segments are delivered to the OSF. Each facility has a set of documentation that defines the delivery requirements and procedures for submitting segments. There are documentation and support requirements that need to be identified. Contact representatives of these facilities for details on how to obtain the documents.

> **NOTE:** MakeInstall versions of segmented applications are submitted for all Windows NT segments. All UNIX segments are delivered as tar tapes (not created using MakeInstall).

Appendix A: Sample Segmentation Descriptor Files

LOGIDB

Descriptor File:	SegName
Mission Application Name:	Logistics Shared Database Segment
Target Operating System:	HP-UX
Comments:	$SEGMENT defined as applicable to the System Administrator Account Group. Eventually, a JLSC or GCSS general user account group should replace this.

```
#============================================================
#
#  LOGIDB Segment SegName
#  Descriptor file.
#
#============================================================

$TYPE:SOFTWARE
$NAME:Logistics Shared Database
$PREFIX:LOGIDB
$SEGMENT:System Administrator:SA:/h/AcctGrps/SysAdm
```

Descriptor File:	ReleaseNotes
Mission Application Name:	Logistics Shared Database Segment
Target Operating System:	HP-UX
Comments:	Format of ReleaseNotes is left up to the discretion of the developer.

```
#============================================================
#
#  LOGIDB Segment ReleaseNotes
#  Descriptor file.
#
#============================================================
```

10/31/96

This is an initial segmentation of the Logistics Shared Database.

This segment requires Oracle 7.3
This segment creates an Oracle tablespace which is 150M in size. In order to create the tablespace, the Installer must know the password to the Oracle 'system' account.

Due to the number of Oracle objects created by this segment, it may take 4 or more hours to completely install the segment.

Descriptor File:	VERSION
Mission Application Name:	Logistics Shared Database Segment
Target Operating System:	HP-UX
Comments:	The time and date were automatically inserted by executing the TimeStamp tool.

```
#============================================================
#
#  LOGIDB Segment VERSION
#  Descriptor file.
#
#============================================================

1.0.0.0 : 11/01/96: 09:31
```

Descriptor File:	SegInfo
Mission Application Name:	Logistics Shared Database Segment
Target Operating System:	HP-UX
Comments:	This segment does not have any binary code and thus can run on all CPUs. Segment has a dependency on Oracle COTS product which is defined under [Requires]. Both PostInstall and DEINSTALL have to execute with root privileges. This is accomplished by listing these files under the [Direct] section.

```
#============================================================
#
#  LOGIDB Segment SegInfo
#  Descriptor file.
#
#============================================================

 [Hardware]
$CPU:ALL
$DISK:4277:150000
$OPSYS:ALL
$TEMPSPACE:10240
$MEMORY:1024

[Security]
UNCLASS

[Requires]
Oracle DB Server Applications:ORAS:/h/COTS/ORAS:1.0.0.4

[Direct]
$ROOT:PostInstall
$ROOT:DEINSTALL
```

Descriptor File:	PostInstall
Mission Application Name:	Logistics Shared Database Segment
Target Operating System:	HP-UX
Comments:	PostInstall executes as a cshrc shell under Unix. PostInstall includes logging into Oracle as the Oracle DBA. Post Install then calls developer generated script files to build the Oracle Tablespace from scratch.

```
#============================================================
#
#   LOGIDB Segment PostInstall
#   Descriptor file.
#
#============================================================

#!/bin/csh -f

set dbffile = $INSTALL_DIR/dbf_file.dat

echo "$INSTALL_DIR/DBS_files/LOGIDBusers.dbf" >! $dbffile

su - oradba -c "setenv DISPLAY $DISPLAY; setenv INSTALL_DIR $INSTALL_DIR;
$INSTALL_DIR/Install/m00mstr.csh"
exit (0)
```

Descriptor File:	FileAttribs
Mission Application Name:	Logistics Shared Database Segment
Target Operating System:	HP-UX
Comments:	This file was created by executing the MakeAttribs tool which is part of the DII COE Developers Toolkit for the Unix environment. Each line represents the desired read/write/execute privileges, group ID and owner for each file under the segment directory.

755:100:1:Scripts
755:100:1:bin
755:100:1:data
755:100:1:man
755:201:102:DBS_files
755:201:102:Install
640:201:102:Install/m00acfn.sql
640:201:102:Install/m00acix.sql
640:201:102:Install/m00actb.sql
640:201:102:Install/m00actc.sql
640:201:102:Install/m00acvw.sql
640:201:102:Install/m00agrl.sql
640:201:102:Install/m00aout.sql
640:201:102:Install/m00drop.sql

Descriptor File:	DEINSTALL
Mission Application Name:	Logistics Shared Database Segment
Target Operating System:	HP-UX
Comments:	This file initiates clean-up associated with the deleting the segments Oracle tablespaces. Implemented in Cshrc shell, the file includes error checking and error messages to the operator.

```
#============================================================
#
#   LOGIDB Segment DEINSTALL
#   Descriptor file.
#
#============================================================


#!/bin/csh -f

set dbffile = `cat $INSTALL_DIR/dbf_file.dat`

su - oradba -c "cd $INSTALL_DIR/Install; svrmgrl < m00drop.sql"

set errors = `grep -c 'ORA-' $INSTALL_DIR/Install/m00drop.out`

if ($errors == 0) then
  rm -f $dbffile
  exit 0
else
  /h/COE/bin/COEMsg \
"ERROR: $error Oracle error(s) were generated while deinstalling\
LOGIDB.  These errors must be fixed before the deinstall can\
continue.  Check /h/LOGIDB/Install/m00drop.out for the errors.\
NOTE:  Do not press the OK button below until the errors have been corrected."
  exit 1
endif
```

Appendix B: Sample Segmentation Descriptor Files

CATII

Descriptor File:	SegName
Mission Application Name:	Control and Analysis Tool
Target Operating System:	HP-UX
Comments:	$SEGMENT defined as applicable to the System Administrator Account Group. Eventually, a JLSC or GCSS general user account group should replace this. Segment type defined as SOFTWARE even though it is a COTS product. The DII COE I&RTS states that if possible, a COTS segment should be segmented as a SOFTWARE type segment instead of a COTS type segment. This was the case here.

```
#============================================================
#
#  CATII Segment SegName
#  Descriptor file.
#
#============================================================

$TYPE:SOFTWARE
$NAME:Control and Analysis Tool
$PREFIX:CATII
$SEGMENT:System Administrator:SA:/h/AcctGrps/SysAdm
```

Descriptor File:	ReleaseNotes
Mission Application Name:	Control and Analysis Tool
Target Operating System:	HP-UX
Comments:	Release notes describe some of the modifications to some of the vendor provided shell scripts that were made to automate the installation process. Some work-arounds that were required to complete the segmentation effort are also described.

#===
#
CATII Segment **ReleaseNotes**
Descriptor file.
#
#===

10/30/96

This is the Control and Analysis Tool.

The bin/unix_script/rgcat.rel shell script was modified by the DISA Team as part of the Segmentation Workshop exercise. The definition of CATLIB was changed to point to /h/CATII/bin.

The SegInfo file points to 5 script files (.cshrc, .login, .profile, .history, .exrc) in the /h/USERS/esales user directory to establish the CATII run-time environment.

CATII requires that the word 'bin' not be included in the executable path to the PDMSS application. Therefore, the executables for PDMSS were put in a 'progs' subdirectory to the PDMSS segment.

CATII also requires that the data files be contained in a 'bin' subdirectory to the directory containing the PDMSS application executables.

Descriptor File:	VERSION
Mission Application Name:	Control and Analysis Tool
Target Operating System:	HP-UX
Comments:	Version number includes the COE formatted version number followed by the vendors own version number. This is recommended any time the COE format version number system is different from the developers.

```
#============================================================
#
#  CATII Segment VERSION
#  Descriptor file.
#
#============================================================
# CATII initial segment  30 Oct 96
1.0.0.0/1.2 : 10/31/96: 10:57
```

Descriptor File:	SegInfo
Mission Application Name:	Control and Analysis Tool
Target Operating System:	HP-UX
Comments:	Segment is for HP platforms executing the HP-UX operating system. The disk space requirement was inserted automatically by the DII COE CalcSpace tool which is part of the DII COE Developers Toolkit. This segment added entries to the /etc/crontab file via the [Community] section of SegInfo. [Comm.deinstall] shows how the /etc/crontab file gets restored upon a DEINSTALL of this segment. [Processes] is used to start a background process at boot time. The files external to the segment directory associated with the segment are listed under [FileList] because it is a COTS product.

```
#============================================================
#  CATII Segment SegInfo
#  Descriptor file.
#============================================================

[Direct]
$ROOT:PostInstall

[Hardware]
$CPU:HP
$OPSYS:HPUX
$DISK:31556
$MEMORY:32000

[Security]
UNCLASS

[Community]
$FILE:/usr/spool/cron/crontabs/root
$APPEND
{
30 11 * * * /bin/rgcat -i k
35 11 * * * /bin/rgcat -i s
}

[Processes]
$BOOT
$PATH:/bin
rgcat -i s

[Comm.deinstall]
$FILE:/usr/spool/cron/crontabs/root
```

```
$DELETE
{
30 11 * * * /bin/rgcat -i k
35 11 * * * /bin/rgcat -i s
}

[FilesList]
$DIRS
/h/CATII
$DIRS
/bin
$FILES
rgcat
$DIRS
/h/USERS/esales
$FILES
.login
.cshrc
.profile
.history
.exrc
```

Descriptor File:	PostInstall
Mission Application Name:	Control and Analysis Tool
Target Operating System:	HP-UX
Comments:	PostInstall includes prompting the operator/installer to get a required software license key from the vendor. It also includes establishing some Xterm parameters, editing a file and calling other script files required for installing this segment.

```
#============================================================
#
#   CATII Segment PostInstall
#   Descriptor file.
#
#============================================================

#!/bin/csh -f

echo "IN the PostInstall for CATII"

mkdir /RGInc

set keyfile = /tmp/pdmss_key.$$

rm -f $keyfile

/usr/bin/X11/xterm -sb -bg yellow -fg black -g +40+40 \
                -title "Get CATII Key License Number" \
                -e $INSTALL_DIR/install/get_key.csh $keyfile

if (! -f $keyfile) then
  /usr/bin/X11/xterm -sb -bg red -fg black -g +40+40 \
                  -title "Error Getting CATII Key License Number" \
                  -e $INSTALL_DIR/install/get_key_error.csh
  goto Continue
endif

set value = `cat $keyfile`

set hostname = `uname -n`

sed -e "s/Hostname=/$hostname=$value/" $INSTALL_DIR/bin/cat2.ini >! /tmp/cat2.tmp
cp /tmp/cat2.tmp $INSTALL_DIR/bin/cat2.ini
rm /tmp/cat2.tmp
rm $keyfile
```

Continue:

```
cp $INSTALL_DIR/bin/unix_script/rgcat.rel   /bin/rgcat

chmod 755 /bin/rgcat

/bin/rgcat -i s

echo "OUT of the PostInstall for CATII"

exit(0)
```

Descriptor File:	DEINSTALL
Mission Application Name:	Control and Analysis Tool
Target Operating System:	HP-UX
Comments:	PostInstall calls a vendor executable to shutdown a deamon.

```
#============================================================
#
#   CATII Segment DEINSTALL
#   Descriptor file.
#
#============================================================

#!/bin/csh -f

echo "IN the DEINSTALL for CATII"

# Shut down TalkMaster
/bin/rgcat -i k

/bin/rm -f /bin/rgcat

echo "OUT of the DEINSTALL for CATII"

exit(0)
```

Descriptor File:	FileAttribs
Mission Application Name:	Control and Analysis Tool
Target Operating System:	HP-UX
Comments:	This file was created by executing the MakeAttribs tool which is part of the DII COE Developers Toolkit for the Unix environment. Each line represents the desired read/write/execute privleges, group ID and owner for each file under the segment directory. This is only page of several pages associated with the file.

775:100:1:bin
554:100:1:bin/BUILD.LOG
755:100:1:bin/app-defaults
755:100:1:bin/app-defaults/bitmaps
554:100:1:bin/app-defaults/bitmaps/rg_about.gif
554:100:1:bin/app-defaults/bitmaps/bug_icon
554:100:1:bin/app-defaults/bitmaps/cat_logo
554:100:1:bin/app-defaults/bitmaps/exit
554:100:1:bin/app-defaults/bitmaps/gray
554:100:1:bin/app-defaults/bitmaps/info
554:100:1:bin/app-defaults/bitmaps/myhand
554:100:1:bin/app-defaults/bitmaps/nle_icon
554:100:1:bin/app-defaults/bitmaps/nle_stipple
554:100:1:bin/app-defaults/bitmaps/radar_pix
554:100:1:bin/app-defaults/bitmaps/rg_busy
554:100:1:bin/app-defaults/bitmaps/vfx_logo
554:100:1:bin/app-defaults/bitmaps/zoom
554:100:1:bin/app-defaults/CatFonts
554:100:1:bin/app-defaults/Deis
554:100:1:bin/app-defaults/Fsx
554:100:1:bin/app-defaults/Nle
554:100:1:bin/app-defaults/Nle.bw
554:100:1:bin/app-defaults/Nle.color
554:100:1:bin/app-defaults/Rgabout
554:100:1:bin/app-defaults/Rgabout.bw
554:100:1:bin/app-defaults/Rgabout.color
554:100:1:bin/app-defaults/Rgcat
554:100:1:bin/app-defaults/Rgcat.bw
554:100:1:bin/app-defaults/Rgcat.color
554:100:1:bin/app-defaults/Vfx
554:100:1:bin/app-defaults/Vfx.bw
554:100:1:bin/app-defaults/Vfx.color
554:100:1:bin/app-defaults/ViewGif
554:100:1:bin/app-defaults/ViewGif.bw
554:100:1:bin/app-defaults/ViewGif.color

Appendix C: Sample Segmentation Descriptor Files

DRS

Descriptor File:	SegName
Mission Application Name:	Discrepancy Reporting System
Target Operating System:	WindowsNT
Comments:	$SEGMENT defined as applicable to the System Administrator Account Group. Eventually, a JLSC or GCSS general user account group should replace this. Segment type defined as SOFTWARE even though it is a COTS product.

```
#=========================================
#
#       DRS Segment SegName
#       Descriptor file.
#
#=========================================

$TYPE:SOFTWARE
$NAME:DRS
$PREFIX:DRS
$SEGMENT:DRS:DRS:\dii\SysAdm
```

Descriptor File:	ReleaseNotes
Mission Application Name:	Discrepancy Reporting System
Target Operating System:	WindowsNT
Comments:	Release notes indicate the developer version number of the software.

```
#============================================================
#
#  DRS Segment ReleaseNotes
#  Descriptor file.
#
#============================================================
```

DRS Software Segment Version 1.1a

Descriptor File:	VERSION
Mission Application Name:	Discrepancy Reporting System
Target Operating System:	WindowsNT
Comments:	TimeStamp is not available for the WindowsNT and thus the date is always entered via an editor.

```
#============================================================
#
#  DRS Segment VERSION
#  Descriptor file.
#
#============================================================
```

1.1.1.0:9/11/96

Descriptor File:	SegInfo
Mission Application Name:	Discrepancy Reporting System
Target Operating System:	WindowsNT
Comments:	Disk space requirement calculated inserted via the editor because CalcSpace is not yet available for the Windows NT environment. Segment has a minimum SegInfo file.

```
#============================================================
#
#  DRS Segment SegInfo
#  Descriptor file.
#
#============================================================

[Hardware]
$CPU:PC
$OPSYS:NT
$DISK:84000
$MEMORY:1024

[Requires]

[Icons]

[Menus]

[Security]
UNCLASS
```

Appendix D: Sample Segmentation Descriptor Files

PDMSS

Descriptor File:	SegName
Mission Application Name:	Program Depot Maintenance Scheduling System
Target Operating System:	HP-UX
Comments:	$SEGMENT defined as applicable to the System Administrator Account Group. Eventually, a JLSC or GCSS general user account group should replace this.

$TYPE:SOFTWARE
$NAME:PDMSS Application
$PREFIX:PDMSS
$SEGMENT:System Administrator:SA:/h/AcctGrps/SysAdm

Descriptor File:	ReleaseNotes
Mission Application Name:	Program Depot Maintenance Scheduling System
Target Operating System:	HP-UX
Comments:	Release notes indicate a dependency on the CATII segment. It also contains information about some directory structure requirements by the segment.

10/30/96

This is the Program Depot Maintenance Scheduling System.

This uses 1 semaphore.

This segment requires the CATII segment to be installed.

CATII requires that the word 'bin' not be included in the executable path to the PDMSS application. Therefore, the executables for PDMSS were put in a 'progs' subdirectory to the PDMSS segment.

CATII also requires that the data files be contained in a 'bin' subdirectory to the directory containing the PDMSS application executables.

Descriptor File:	VERSION
Mission Application Name:	Program Depot Maintenance Scheduling System
Target Operating System:	HP-UX
Comments:	Date and time inserted by TimeStamp tool..

PDMSS initial segment 30 Oct 96
1.0.0.0 : 10/31/96: 10:57
1.0.0.1

Descriptor File:	SegInfo
Mission Application Name:	Program Depot Maintenance Scheduling System
Target Operating System:	HP-UX
Comments:	Dependency on CATII segment is identified under [Requires]. [Direct] indicates that the application will accessible via X-Term session and that PostInstall must be executed as root. He crontab file gets updated with a placeholder for an FTP program via [Community]. [comm.deinstall] will undo what [Community] does upon a DEINSTALL . A new group is defined under [COEServices]

```
# SegInfo for PDMSS Segment

[Direct]
$ROOT:PostInstall
$REMOTE:XTERM

[Hardware]
$CPU:HP
$OPSYS:HPUX
$DISK:193112
$MEMORY:32000

[Requires]
Control and Analysis Tool:CATII:/h/CATII:1.0.0.0

[Security]
UNCLASS

[Community]
$FILE:/usr/spool/cron/crontabs/root
$APPEND
{
#Placeholder for FTP transfer
}

[Comm.deinstall]
$FILE:/usr/spool/cron/crontabs/root
$DELETE
{
#Placeholder for FTP transfer removal
}

[COEServices]
$GROUPS
C130:212
```